UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE

PAR
DIDIER FÉLIX EKE

MÉTHODE D'IDENTIFICATION EN VIRGULE FIXE D'UN MODÈLE NON LINÉAIRE BASÉ SUR LES ALGORITHMES GÉNÉTIQUES

MAI 2008

Résumé

Le filtrage adaptatif est présent en traitement du signal dans de nombreuses applications, tels que : l'identification de système, l'égalisation, l'annulation d'écho ou encore la réduction d'interférence. Les méthodes adaptatives en traitement du signal visent l'adaptation automatique des opérateurs de traitement aux propriétés statistiques des signaux et des systèmes, ainsi que l'adaptation à leurs variations dans le temps. Pour les systèmes de communications sans fil numériques modernes, l'identification de canaux (systèmes) est une application essentielle, souvent largement souvent utilisée. La connaissance des paramètres du canal de communication permet de faire une très bonne estimation des données transmises et ainsi d'avoir un taux d'erreur presque nul. Pour beaucoup de systèmes sans fil, le canal considéré peut être modélisé comme un filtre à réponse impulsionnelle finie (FIR) à temps variant.

Compte tenu de la mobilité dans le temps du canal, il est utile d'avoir un algorithme qui peut identifier les paramètres du canal de manière adaptative. Les algorithmes adaptatifs tels que : l'algorithme du gradient stochastique (LMS - *Least Mean Square*) et ses variantes (ex: LMS normalisé) et l'algorithme des moindres carrés récursifs (RLS – *Recursive Least Square*) sont des méthodes connus et qui sont à même de résoudre le problème qu'est l'identification d'un système. Les méthodes citées plus haut possèdent les faiblesses suivantes : le LMS et sa variante le NLMS possèdent un taux de convergence lent. Le RLS possède un taux de convergence plus rapide, cependant son implémentation est plus

complexe et aussi plus coûteuse, cela à cause de sa matrice de covariance qui exige un grand nombre d'opérations arithmétiques. Les autres faiblesses de ces méthodes résident dans leur incapacité à identifier des systèmes non linéaires ainsi que leur incapacité à fonctionner avec très peu de bits lors d'une implémentation en VLSI (*Very Large Scale Integration*).

Ce mémoire présente une méthode d'identification robuste des systèmes basée sur les algorithmes génétiques (AG). Contrairement aux méthodes de filtrage mentionnées plus haut la méthode basée sur les AG possède les avantages suivants : elle opère aussi bien dans les systèmes linéaires que non linéaires, elle est capable de procéder à l'identification de système même avec très peu de bits, il n'est pas nécessaire de procéder à une étude de quantification complète, car la méthode intègre dans son mode de fonctionnement un parallélisme implicite, son taux de convergence est assez rapide, car elle s'inspire de la théorie de l'évolution de Darwin.

Ce mémoire présente une étude complète de la méthode des AG, une étude comparative entre les méthodes classiques nommées plus haut et la méthode basée sur les AG développée dans le cadre de ce travail. Pour compléter l'objectif de la recherche, une implémentation de la méthode basée sur les AG sous Simulink, est proposée dans la perspective de futurs travaux qui pourraient être la conversion des blocs Simulink en bloc Xilinx afin de faire une synthèse de l'architecture.

Remerciements

Je tiens tout d'abord à remercier le Professeur Éloi Ngandui, qui fut ma source d'inspiration et qui motiva également mon orientation académique vers le génie électrique. Je tiens également à remercier mon directeur de recherche, le Professeur Daniel Massicotte. Mes remerciements vont aussi à ma famille, plus particulièrement à ma mère Marie-Louise Ngadjui, qui m'a toujours soutenu et encouragé. J'adresse aussi mes plus sincères remerciements à mes collègues Élie et Patrick pour leur support technique et moral.

Table des matières

Liste des tableaux

Liste des figures

Liste des symboles

AG	Algorithmes génétiques
ARMA	Autoregressive moving average
BER	Bit error rate
W	Taille de la fenêtre de calcul de l'erreur
e	Erreur
f	fonction sélective
F	fonction sélective de toute la population
FIR	Finite impulse response
EKF	Extended Kalman Filter
HPAs	High power amplifiers
LMS	Least mean square
m	Taille de la population
M	Nombre de coefficients du filtre
MSE	Mean square error
NLMS	Normalize least mean square

p_c	Pourcentage de recombinaison de la population
P_j	Probabilité de sélection pour chaque chromosome
p_m	Pourcentage de mutation de la population
Pop_j	Individu de la population
q_j	Probabilité cumulative pour chaque chromosome
RLS	Recursive least square
VLSI	Very large scale integration
$w(n)$	Vecteur des poids du filtre
ε	Critère de performance du système
D	Délai

Chapitre 1 - Introduction

Depuis les années 60, le filtrage adaptatif a suscité un engouement et un développement sans précédent. Ce développement du filtrage adaptatif est né de l'essor du traitement numérique, de la croissance soutenue de la puissance des processeurs de traitement qui permettent la mise en œuvre en temps réel, d'algorithmes de plus en plus complexes et qui vont à des cadences de plus en plus élevées [1].

Suite à cet essor du filtrage adaptatif, de nombreuses applications telles que : l'identification des systèmes [2], l'égalisation de canaux [3], la modélisation des systèmes [4], la résolution de problème inverse [5] ont vu le jour. Des applications mentionnées plus haut, certaines comme l'identification des systèmes, sont un problème important dans des domaines aussi variés que : le traitement numérique du signal [6], le contrôle de procédé [7], ou encore les communications [8], [9]. L'identification de système est réalisée par l'ajustement de paramètres pour un modèle donné jusqu'à ce que ses sorties, pour une entrée particulière, coïncident autant que possible avec la sortie mesurée du système identifié pour la même entrée.

L'identification des systèmes linéaires est effectuée depuis des décennies à l'aide des algorithmes du gradient stochastique [10] tels que : le *LMS (Least Mean Square), sa* méthode normalisé *NMLS (Normalized LMS).* L'identification s'est également beaucoup effectuée à l'aide de méthode de type récursif comme le *RLS (Recursive Least Square).* Cependant aujourd'hui, à cause de la complexité des systèmes, ces derniers deviennent de

plus en plus non linéaires [11], [12]. Compte tenu de l'usage de plus en plus grand de modèles non linéaires dans les systèmes réels, de nombreuses méthodes de résolution pour l'identification de systèmes non linéaires, tels que les réseaux de neurones [13] et le filtre de Kalman étendu [14] ont été développées.

Les systèmes numériques ont fait leur apparition il y a plusieurs décennies. Dans ces derniers, les opérations arithmétiques sont effectuées en virgule fixe. L'exécution d'opération en virgule fixe présente les avantages suivant dans les implémentations en Intégration à très grande échelle (VLSI – Very Large Scale Integration) : une faible consommation de puissance, une surface d'intégration réduite et une vitesse de calcul élevée. Cependant, dans l'identification, les caractéristiques de traitement (convergence, précision, erreur asymptotique, stabilité) sont affectées par le nombre de bits employés pour calculer les coefficients du filtre [15].

1.1 Problématique

L'identification des systèmes regroupe à la fois l'identification des systèmes linéaires et celui des systèmes non linéaires. L'identification des systèmes linéaires est connue et maîtrisée depuis fort longtemps, cela à travers divers algorithmes dédiés au filtrage adaptatif.

Les algorithmes comme le *LMS*, le *NMLS* et le *RLS* permettent d'obtenir des résultats satisfaisants lors de l'identification de système linéaire. Toutefois lorsque le système devient non linéaire, ils deviennent inefficaces, incapables de bien identifiés le système et dans certains cas si le degré de non linéarité du canal est élevé ils deviennent inopérants. Ces algorithmes possèdent d'autres faiblesses comme la vitesse de convergence qui est lente pour le *LMS* et ses variantes. Dans le cas du *RLS*, la vitesse de convergence est rapide,

l'erreur asymptotique est relativement faible, cependant son implémentation en *VLSI* est complexe et coûteuse. L'autre faiblesse majeure inhérente aux algorithmes d'adaptations mentionnés plus haut est leur incapacité à fonctionner avec une faible longueur binaire des coefficients lors d'une implémentation arithmétique entière ou à virgule fixe. Cette incapacité provient d'une instabilité causée par la boucle de récurrence de ces méthodes.

Compte tenu des lacunes imposées par le *LMS* et ses variantes ainsi que le *RLS*, ce travail propose une méthode robuste d'identification des systèmes basée sur les algorithmes génétiques (AG). À l'opposé des méthodes de filtrage mentionnées plus haut la méthode d'identification basée sur les AG possède les avantages suivants :

- elle opère aussi bien dans les systèmes linéaires que non linéaires.
- elle est capable de procéder à l'identification de système même avec peu de bits.
- pour la représentation des nombres entiers, il n'est pas nécessaire de procéder à une étude de quantification complète, car la méthode intègre dans son mode de fonctionnement un parallélisme implicite.
- son taux de convergence est assez rapide.

1.2 Objectifs

L'objectif de ce mémoire est d'effectuer une étude de la méthode basée sur les AG appliquées à l'identification des paramètres de systèmes linéaires et non linéaires. Les sous objectifs sont:

- une étude comparative entre les méthodes classiques nommées plus haut et la méthode adaptative basée sur les AG.

- effectuer une étude comparative de la complexité des diverses méthodes étudiées.
- effectuer une implémentation de la méthode basée sur les AG dans l'environnement Simulink en vue d'une implémentation sur FPGA.

1.3 Méthodologie

Une revue approfondie de la littérature, permettra de bien comprendre le fonctionnement des algorithmes génétiques, leurs différentes applications, leur implémentation en VLSI. Cette recherche littéraire sur les AG fournira les outils nécessaires afin d'être capable de faire les bons choix en ce qui concerne les éléments de la méthode tels que : le choix du codage des chromosomes, des opérateurs génétiques, du critère d'arrêt et de la fonction sélective. La recherche bibliographique permettra également de donner un aperçu de l'état de la problématique de l'identification, de faire le choix du canal à identifier qui devra présenter une non linéarité et une complexité assez élevées afin de montrer l'efficacité de notre méthode.

Le détail de la mise en œuvre des AG appliqués à l'identification du canal choisi sera présenté, à savoir les études et les simulations qui ont permis le choix de certains paramètres de la méthode. Une étude comparative entre les AG et les méthodes classiques telles que le *LMS*, le *NLMS* et le *RLS* sera effectuée. Cette comparaison se fera à partir des courbes de convergence de la moyenne carrée de l'erreur (MSE), convergence de la fonction de coût, dans les cas linéaires et non linéaires et pour diverses longueurs des coefficients binaires lors de l'étude de quantification. Les comparaisons seront également présentées dans des tableaux où les valeurs réelles des paramètres et celles obtenues avec les algorithmes seront commentées.

Les résultats décrits dans le paragraphe plus haut, seront obtenus à l'aide du logiciel MATLAB. Une implémentation de l'AG développé sera effectuée dans Simulink.

1.4 Organisation du mémoire

Le filtrage adaptatif, ainsi que les algorithmes génétiques seront traités dans le chapitre 2. La première section du chapitre 2 abordera de façon générale le filtrage adaptatif, les sections 2.2 et 2.3 aborderont respectivement le filtrage linéaire et le filtrage non linéaire. Dans la section 2.2, des sous-sections présenteront les équations et les étapes de réalisation du *LMS,* du *NLMS* et du *RLS.* Les sous-sections de la section 2.3 seront consacrées à la présentation des causes de non-linéarité dans les systèmes ainsi que certains algorithmes de traitement pour le filtrage non linéaires. La quatrième partie du chapitre 2 est liée à la représentation des nombres binaires et à l'effet de quantification.

L'étude générale des AG commence à la section 2.5. La section 2.5 du chapitre 2 fait un historique et présente le principe de base des AG. Le vocabulaire propre aux AG est présenté à la section 2.6. Les sections 2.7, 2.8, 2.9, 2.10, 2.11 abordent respectivement les sujets suivant : la création de la population, l'évaluation de la population, la sélection, les opérateurs génétiques, les avantages et les inconvénients des AG.

Le chapitre 3 présente les détails de l'application des AG à l'identification d'un système de type ARMA. La définition de l'identification est faite à la section 3.1, tandis que celle du canal ARMA (linéaire, non linéaire) est effectuée à la section 3.2. La section 3.3 s'occupe des aspects suivants : la création de la population, l'évaluation de la population, la sélection avec la roulette, la recombinaison, la réévaluation de la population, la mutation, la réévaluation de la population et réinsertion.

La Chapitre 4 est essentiellement axé sur la présentation des résultats. Les sections 1 et 2 présente le détail du choix des paramètres des différentes méthodes étudiées. Les sections 4.3 à 4.6 s'attardent respectivement sur : les critères de comparaison des méthodes, les résultats pour le canal linéaire, les résultats pour le canal non linéaire, l'étude de complexité et pour terminer une discussion.

Le chapitre 5 aborde l'implémentation de la méthode des AG sous Simulink. L'explication de mise en œuvre de la méthode est effectuée à la section du chapitre, tandis que les résultats sont présentés dans la seconde section du dit chapitre.

La conclusion générale du mémoire est réalisée au chapitre 6, suivis de la bibliographie et de l'annexe.

Chapitre 2 - Algorithmes adaptatifs et algorithmes génétiques

2.1 Principe du filtrage adaptatif

Le filtrage adaptatif ainsi que ses méthodes d'application ont connu un développement considérable depuis les années 60. Ce développement du filtrage adaptatif est né de l'essor du traitement numérique, de la croissance soutenue de la puissance des processeurs de traitement qui permettent la mise en œuvre en temps réel, d'algorithmes de plus en plus complexes et qui vont à des cadences de plus en plus élevées [1].

Les méthodes adaptatives en traitement du signal, ont pour objectif : l'adaptation des outils de traitement aux propriétés statistiques des signaux et des systèmes, ainsi que l'adaptation à leurs fluctuations dans le temps. Il s'agit donc d'un mélange bien équilibré, entre la stationnarité et la non stationnarité. La stationnarité permet de maintenir de façon permanente, dans le temps les propriétés statistiques, grâce auxquelles sont éliminées ou tout au moins réduites les fluctuations purement aléatoires. La non stationnarité, est la variation lente ou rapide, au cours du temps des propriétés statistiques, sans lesquelles, il n'y aurait nul besoin d'adaptation. En l'absence de fluctuation des signaux et systèmes, le filtre optimal pourrait être calculé une seule fois. Les filtres peuvent être classifiés comme étant linéaires ou non linéaires.

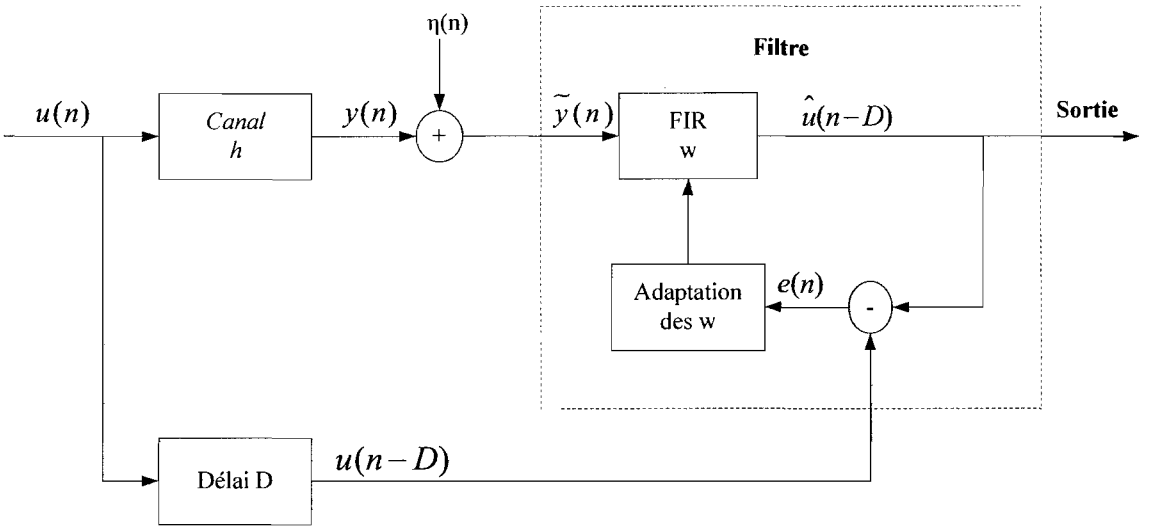

Figure 2.1 Filtrage adaptatif

2.2 Filtrage linéaire

Un système ou filtre est dit linéaire, si la sortie est une fonction linéaire de l'entrée. Les filtres sont conçus en optimisant les coefficients ou poids du filtre en fonction de critères spécifiques. La moyenne des erreurs aux carrées ou *mean square error (MSE)* est souvent utilisée comme critère d'optimisation. Avec ce critère, le filtre est optimisé pour réduire au minimum la valeur de la moyenne carrée de l'erreur du signal. L'erreur du signal est définie comme la différence entre le signal estimé ($\hat{u}(n-D)$) et le signal désiré ($u(n-D)$) du filtre (Figure 2.1). Dans un environnement stationnaire, le filtre de Wiener est la solution linéaire dite optimale au sens *MSE* au problème de filtrage. Le filtre de Wiener assume la connaissance de certains paramètres statistiques, en réalité ces derniers peuvent être inconnus. Pour surmonter cette limitation, on utilise un filtre adaptatif qui estime périodiquement, le plus souvent à chaque échantillon, les coefficients du filtre selon un algorithme choisi. Après un nombre suffisant d'itérations, l'algorithme, idéalement,

converge vers une solution optimale au sens de la fonction de coût MSE minimum. Au cours des années, beaucoup d'algorithmes d'adaptation ont été développés pour concevoir les filtres adaptatifs. Le *Least mean square (LMS),* le *recursive least squares (RLS)* et leurs nombreuses variantes sont parmi les méthodes les plus largement étudiées. Ces techniques de filtrage ont été grandement utilisées et appliquées à la résolution des problèmes tels que : l'identification des systèmes [2], (figure 2.4), l'égalisation de canal en communication [3] (figure 2.3), la soustraction du bruit (figure 2.2), l'annulation d'écho et la réduction d'interférence.

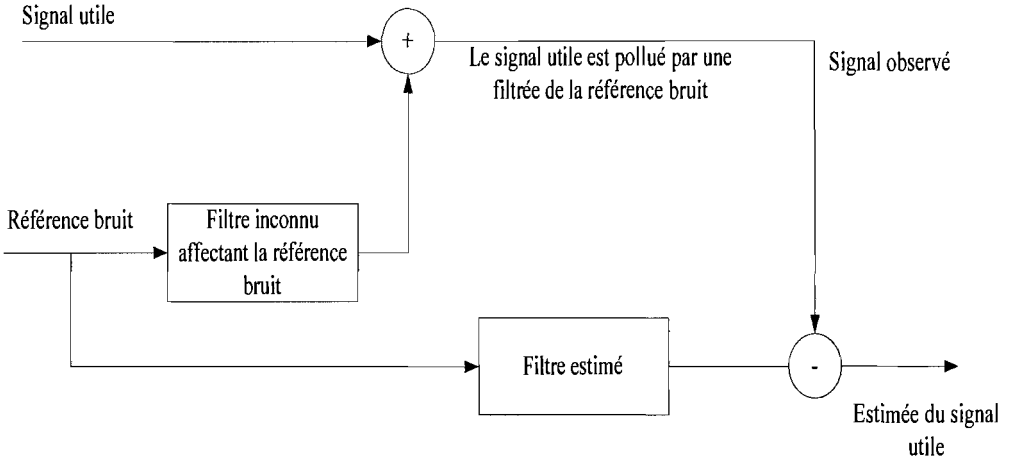

Figure 2.2 Soustraction de bruit

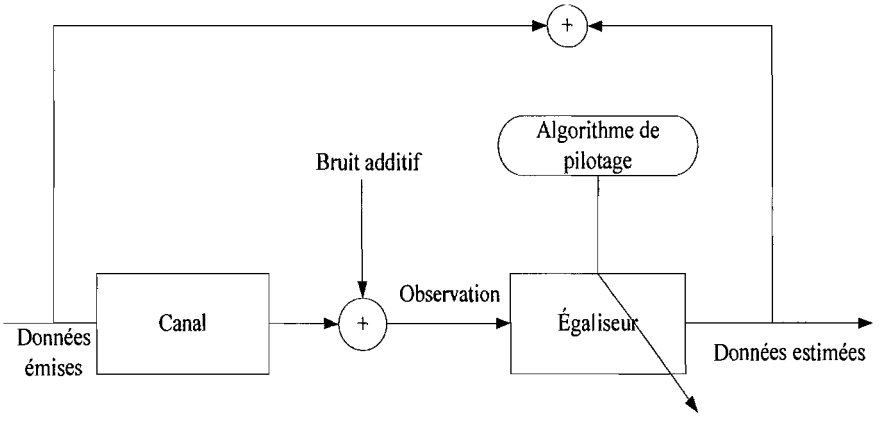

Figure 2.3 Égalisation

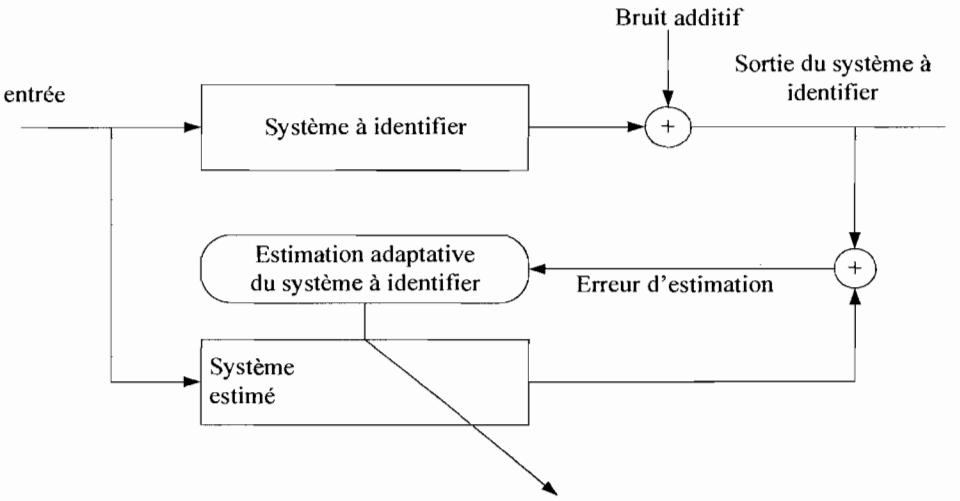

Figure 2.4 Identification

2.2.1 Méthode du gradient stochastique (LMS – least mean square)

Il s'agit d'une des méthodes les plus populaires dans l'industrie pour le calcul des coefficients d'un filtre à réponse impulsionnelle finie (FIR – Finite Impulse Response). Cette méthode consiste en deux étapes fondamentales :

1. *Étape de filtrage :* implique le calcul des données de sortie d'un filtre transversal à partir d'un choix des poids initiaux

$$\hat{u}(n) = w(n)^T \tilde{y}(n) \tag{2.1}$$

et ensuite l'estimation de l'erreur en comparant la sortie du filtre avec la sortie désirée

$$e(n) = u(n-D) - \hat{u}(n-D) \tag{2.2}$$

2. *Étape d'adaptation :* procède à la mise à jour des poids du filtre en fonction de l'estimation de l'erreur

$$w(n+1) = w(n) + \mu_{LMS} e(n) \hat{u}(n) \tag{2.3}$$

Comme présenté à la figure 2.1, cette méthode prend en entrée le signal $\tilde{y}(n)$ reçu après le passage dans un système ayant une impulsionnelle $h(n)$ et additionné d'un bruit $\eta(n)$. n représente ici l'instant d'échantillonnage d'un signal échantillonné aux temps nT, avec T correspondant à la période d'échantillonnage. Le signal ainsi obtenu passe dans un filtre linéaire FIR, selon 2.1, dont les coefficients sont mis sous forme du vecteur colonne qui suit : $w(n)=[w_1(n), w_2(n), ..., w_N(n)]^T$. La sortie estimée de la séquence transmise, notée $\hat{u}(n-D)$, est comparée avec la séquence émise désirée $u(n-D)$, (2.2). L'erreur qui en résulte est utilisée afin de mettre à jour les coefficients $w(n)$ de manière à minimiser l'erreur quadratique moyenne. Ainsi après un certain nombre d'itérations, l'erreur doit tend vers 0, sans toute fois ne jamais l'atteindre. D représente le délai ajouté à $y(n)$ afin de compenser le retard du passage de $u(n)$ dans $h(n)$. μ_{LMS} est appelé, le pas de convergence de la méthode *LMS*, ce paramètre ajustable empiriquement, permet de faire un compromis entre la stabilité la vitesse de convergence et l'erreur asymptotique de la méthode *LMS*. Afin d'assurer la convergence du 1er ordre, il est nécessaire que le pas de convergence soit compris dans l'intervalle ci-dessous:

$$0< \mu_{LMS} < 2/ \lambda$$

avec λ représentant l'auto corrélation des observations, on choisit généralement μ_{LMS} inférieur à 1 puisque son intervalle est fonction des valeurs propres du systèmes. Étant donné que les caractéristiques stochastiques de $\tilde{y}(n)$ varie à chaque échantillon, il est difficile d'établir μ_{LMS}. Afin de rendre l'algorithme moins dépendant des caractéristiques stochastiques des signaux le *NLMS* a été proposé.

2.2.2 *Méthode Normalisée du gradient stochastique (NLMS - Normalized LMS)*

La mise en œuvre du *NLMS* est régie par les mêmes étapes et les mêmes équations que le *LMS*. La différence se situe au niveau de la mise à jour des poids selon la formulation suivante :

$$\mathbf{w}(n+1) = \mathbf{w}(n) + (\mu_{NLMS} e(n)\hat{u}(n)) / (\|u(n)\|^2) \tag{2.4}$$

Le paramètre μ_{LMS} est défini dans l'intervalle $0 < \mu_{NLMS} < 2$. Ce dernier ne dépendant pas des valeurs propres du système, il ne varie donc que très peu. Tant et aussi longtemps que μ_{NLMS} se trouve dans son intervalle de définition, la stabilité ainsi que la convergence vers les solutions optimales sont plus aisées.

2.2.3 *Méthode récursive des moindres carrés (RLS – recursive least squares)*

Le *RLS* est un algorithme d'une complexité plus grande que le *LMS*, cependant il fournit une convergence plus rapide, une erreur asymptotique plus faible et est peu sensible aux variations des valeurs propres du système. Les étapes de sa réalisation comprennent celles citées plus haut pour le *LMS,* avec comme différences majeures : la mise à jour des poids et la présence du gain. L'équation de mise à jour des poids s'écrit comme suit :

$$\mathbf{w}(n+1) = \mathbf{w}(n) + Gconj(e(n)) \tag{2.5}$$

avec le gain G qui se définit comme suit:

$$G(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)u(n)}{1+\lambda^{-1}u^T(n)\mathbf{P}(n-1)u(n)} \tag{2.6}$$

Le calcul de G est effectué à l'aide de la matrice de covariance qui se définit selon :

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}G(n)u^{T}(n)\mathbf{P}(n-1) \tag{2.7}$$

où λ est le facteur d'oubli compris entre 0 et 1. La matrice de covariance est initialisée de la manière suivante :

$$\mathbf{P}(0) = \beta^{-1}\mathbf{I} \tag{2.8}$$

Avec *I,* qui est la matrice identité et β une constante positive très faible.

2.3 Filtrage non linéaire

Les filtres ou systèmes sont dits non linéaires dès lors qu'ils n'obéissent pas au principe de superposition [1]. Les méthodes conventionnelles de filtrage sont inefficaces en présence de non-linéarité. Quelques cas pratiques où la non linéarité est communément rencontrée en traitement de signal sont cités ci-dessous:

- Amplificateurs de haute puissance (*high power amplifiers* - HPAs) [16]: dans les communications sans fil, pour un rendement de puissance élevé, les HPAs sont amenés proches de la saturation (ex: communications satellites). Les HPAs génèrent des amplitudes non linéaires et de la distorsion de phase lorsqu'ils sont proches de la saturation. Cet état à proximité de la saturation cause une dégradation de la performance du taux d'erreur binaire (BER – *Bit Error Rate*) et introduit aussi de l'interférence sur les canaux adjacents aux systèmes opérant dans des bandes de fréquences voisines.

- Canaux optiques [35]: les récepteurs de fibre optique souffrent de plusieurs sources de non linéarité, incluant les photos détecteurs qui convertissent la lumière incidente

en photo courant, la dépendance en intensité de l'index de réfraction de la fibre est amplifiée par les émissions spontanées qui ont une distribution non gaussienne.

- Haut-parleur [36]: les haut-parleurs génèrent de la non linéarité qui dégrade la qualité de l'audio. Les sources de non linéarité sont les non linéarités contenues dans le système de suspension ainsi que la non homogénéité du flux de densité.

- Autres: les systèmes de commandes dynamiques non linéaires, les communications sous-marines et bien d'autres.

Plusieurs méthodes ont été développées depuis des décennies afin de comprendre et résoudre les problèmes de non-linéarités. Les méthodes telles que le filtre de Kalman étendu et le filtre de Volterra seront abordées dans la suite de ce chapitre, tandis que les algorithmes génétiques seront détaillés dans le chapitre 3.

2.3.1 Filtre de Kalman étendu

Le filtre de Kalman [17] est une des méthodes de filtrage adaptatif le plus repondu. Il a été développé à l'origine par Kalman en 1960 pour le temps discret, puis repris en 1961 par Kalman-Bucy pour le temps continu. Le filtre de Kalman est une forme récursive du filtre optimal de Wiener. Le filtre de Kalman original est utilisé pour les systèmes linéaires lorsque les bruits sont additifs et gaussiens. Afin d'être appliqué aux systèmes non linéaires, le modèle original a été étendu. Le filtre de Kalman étendu (EKF – *Extended Kalman Filter*) permet de développer des modèles linéaires pour des systèmes non linéaires [18]. EKF a été employé pour la résolution de systèmes non linéaires dans applications tels que : l'égalisation [19] ou l'identification de canaux.

2.3.2 Filtre de Volterra

La théorie de Volterra [20] a été développée par *Vito Volterra.* Il a étendu l'idée de la convolution linéaire pour prendre en compte les modèles non linéaires. Les filtres de *Volterra* caractérisent un filtre non linéaire à travers un ensemble de coefficients appelés les poids de *Volterra.* Le nombre de poids de *Volterra* se développe avec l'augmentation modérée de l'ordre et de la mémoire du filtre.

2.4 Effets de quantification

2.4.1 Définitions

La représentation réalisée à la figure 2.5 constitue un mot de (n+b+1) chiffres binaires, appelés bits [21]. La base est 2, car lorsqu'on parle de binaire, il s'agit de deux chiffres. Les deux chiffres binaires ou bits, sont le 1 et le 0. On a donc :

$$a_i \in \{0,1\}, \quad i \in \{n, n-1, ..., 1, 0, -1, ..., -b\}$$

Pour chaque chiffre, on associe une puissance de 2, égale à sa position. Le bit situé à l'extrême gauche est le bit de signe, les n bits suivants sont réservés à la partie entière, tandis que les b derniers bits correspondent à la partie fractionnaire.

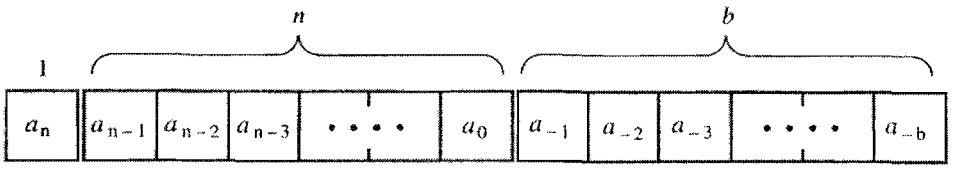

Figure 2.5 Représentation d'un nombre en virgule fixe

Au niveau de la représentation dite *par grandeur et signe* d'un nombre x, les (n+b) derniers bits correspondent au module tandis que l'on convient que $a_n = 0$ pour un nombre

positif, et a_n=1 pour un nombre négatif. On peut donc écrire un nombre x quantifié sur n+b+1 bits :

$$x = (-1)^{a_n} \sum_{i=-b}^{n-1} 2^i a_i \tag{2.9}$$

Deux valeurs représentées successivement, diffèrent d'une quantité appelée *pas de quantification,* et notée q. On a donc :

$$-2^n + q \le x \le 2^n - q, \quad q = 2^{-b} \tag{2.10}$$

La représentation *par complément à 2* est celle qui est la plus couramment utilisée, car elle facilite beaucoup les opérations arithmétiques binaires. De ce fait :

- La représentation d'un nombre positif demeure la même que celle décrite plus haut.
- Dans le cas d'un nombre négatif, le complément de son module à 2^n, soit $2^n - |A|$, est représenté dans la zone réservée au nombre et on pose a_n= 1. avec $|A|$ correspondant à la valeur absolue du module de x.

Suite à cette convention, les opérations d'addition et de changement de signe sont réalisées sur l'ensemble du terme, sans discrimination entre bit de signe et bit de nombre. Ainsi selon la règle énoncée, on a :

$$\begin{aligned} x &= \sum_{i=-b}^{n-1} 2^i a_i & x \ge 0 \\ &= -|x| = -2^0 + \sum_{i=-b}^{n-1} 2^i a_i & x < 0 \end{aligned}, \tag{2.11}$$

Alors que l'expression suivante est générale :

$$x = -2^n a_n + \sum_{i=-b}^{n-1} 2^i a_i \tag{2.12}$$

L'intervalle de représentation est borné par :

$$-2^n \leq x \leq 2^n - q\,,\ q = 2^{-b} \tag{2.13}$$

Pour effectuer le changement de signe d'un nombre représenté en complément à 2, on peut utiliser l'une des techniques suivantes :

- On réalise le complément de tous les bits et on ajoute q.
- On réalise le complément de tous les bits situés à la gauche du bit significatif de plus faible poids.

2.4.2 Erreur de quantification

La représentation d'une grandeur de type virgule flottante, en virgule fixe occasionne une erreur appelée, *erreur de quantification.* La quantification par arrondi (*rounding - RD))* consiste à choisir la valeur représentable *Q(x),* la moins éloignée de la valeur idéale *x*. En outre lorsque la valeur *x* est située à entre deux valeurs quantifiées, on attribue à *x,* la valeur supérieure *(up-rounding).*

L'*erreur d'arrondi* (erreur de quantification) s'écrit e_{RD} :

$$e_{RD} = Q_{RD}(x) - x \tag{2.14}$$

En l'absence de dépassement, on a:

$$-q/2 < e_{RD} \leq q/2 \tag{2.15}$$

La règle illustrée à l'équation (2.15), correspond à l'arrondi du complément à 2. Ce mode de quantification est illustré à la figure 2.6.

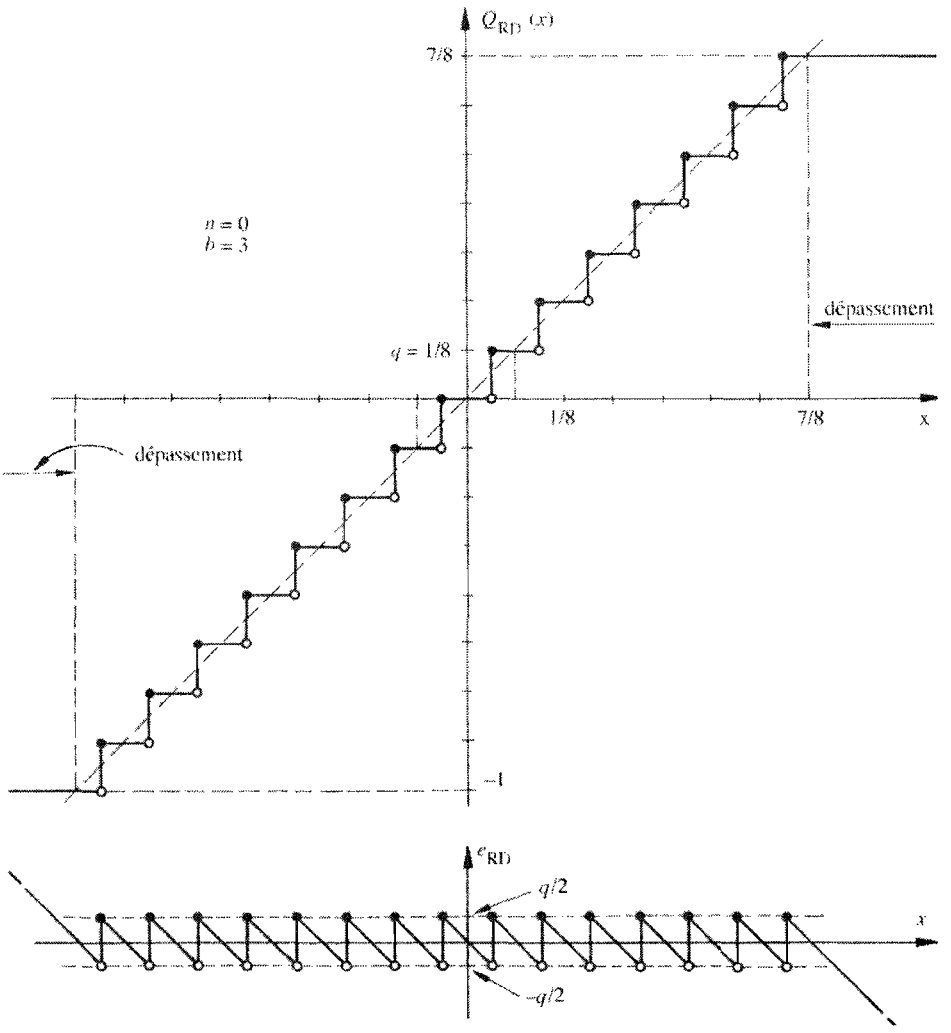

Figure 2.6 Quantification par arrondi de la représentation par complément à 2 [1]

A titre d'exemple d'effet de quantification, les courbes présentant les résultats en virgule flottante, sur 16 et 24 bits d'un RLS pour l'identification d'un système de type ARMA sont présentés à la figure 2.7.

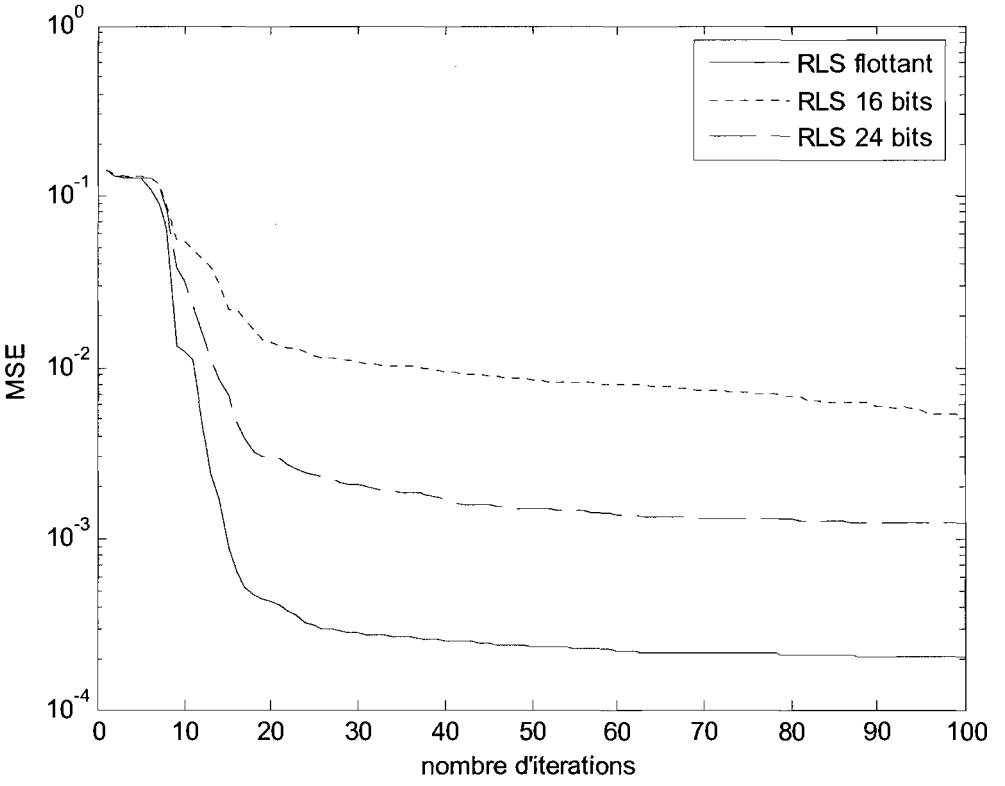

Figure 2.7 Application du RLS à l'identification d'un système de type *ARMA* pour diverses longueurs de bits

Les résultats sur 16 et 24 bits présentés à la figure 2.7 diffèrent de celui en virgule flottante à cause des erreurs lors de la quantification. En effet, plus la longueur la longueur du mot diminue, plus il y a accumulation des erreurs provenant des calculs des équations du RLS (2.5) à (2.8). La matrice de covariance $\mathbf{P}$, demande beaucoup de précision et la quantification de ses coefficients devient catastrophique lorsque la représentation binaire est insuffisante.

2.5 Principe de base des algorithmes génétiques

Les algorithmes génétiques (AG) font partie de la famille des algorithmes évolutionnaires. Ce sont des algorithmes d'optimisation, ils ont été initialement introduits par John Holland en 1975 [22]. Ils ont ensuite connu une exploration intensive grâce aux travaux de Goldberg [23] et De Jong [24-25]. Les algorithmes génétiques s'inspirent du concept de l'évolution naturelle élaboré par Charles Darwin : « appliquer à une population des transformations afin de produire une population mieux adaptée ». Les AG permettent la résolution d'une grande variété de problèmes d'optimisation à l'exemple de l'organisation d'un emploi de temps de travail basée sur les AG [26] ou encore la recherche de design optimal de suspension pour les véhicules grâce aux AG [27]. Dans le traitement adaptatif des signaux, les AG ont été appliqués à l'estimation du retard d'un signal [28], l'adaptation des poids pour un réseau de neurones [29] et l'estimation des paramètres pour des filtres adaptatifs linéaires et non linéaires [30].

Les AG sont différents des techniques d'optimisation conventionnelles. Ils effectuent des recherches aléatoires dans un ensemble spécifié de solutions possibles, cela dans le but de trouver la meilleure solution, en se basant sur des critères de qualité ou de satisfaction. La mise en œuvre de la méthode débute par la création d'un ensemble de solutions aléatoires appelé population. Chaque individu de la population est appelé chromosome, chacun des individus représente une solution potentielle au problème à résoudre. Un chromosome est une chaîne de longueur finie, constituée de symboles appartenant à un alphabet fini. Il est d'habitude, mais pas toujours une chaîne binaire. Les chromosomes évoluent à travers des itérations successives, appelées générations. Au cours de chaque génération, les chromosomes sont évalués en se basant sur les fonctions sélectives. Une nouvelle

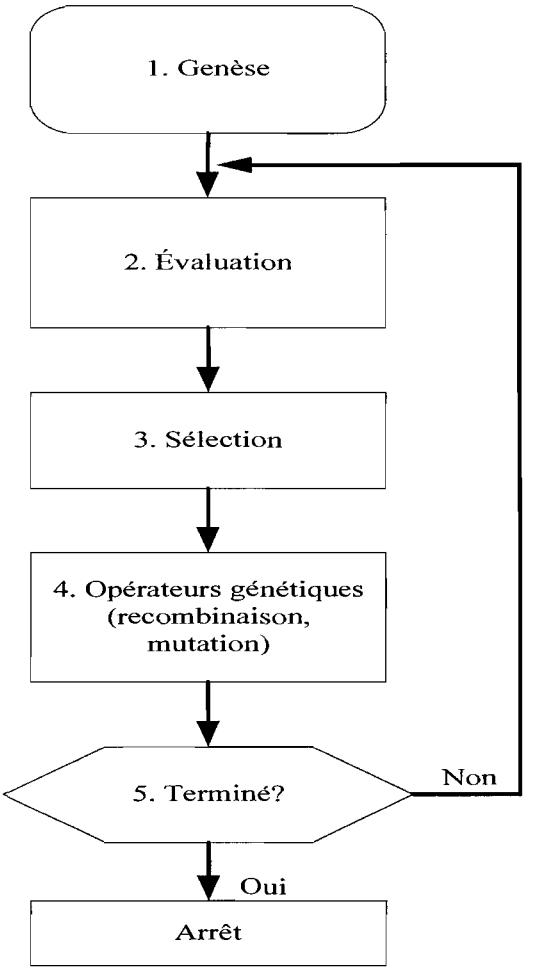

Figure 2.8 Structure générale d'un Algorithme Génétique

population est obtenue en sélectionnant au regard des résultats de la fonction sélective, ceux des chromosomes qui sont les plus performants et en rejetant ceux qui sont les moins performants afin de garder la taille de la population constante. Les chromosomes les plus performants au sens de la fonction sélective ont une plus grande probabilité d'être sélectionnés. Dans le but de créer les prochaines générations, les futurs chromosomes dénommés *enfants,* sont issus soit de la fusion de deux chromosomes de la présente génération en utilisant la recombinaison, soit en altérant légèrement un chromosome en faisant usage de la mutation. Après plusieurs générations, l'algorithme converge vers le

meilleur individu, représentant la solution optimale selon la fonction sélective posée. Les étapes générales d'un algorithme génétique (Figure 2.7) se décrivent comme suit :

Début

1. Création aléatoire d'un ensemble initial de chromosome (genèse) :
   - Population initiale.
2. Évaluation de chaque chromosome en terme "d'ajustement" (fitness) avec la fonction sélective posée.
   - Évaluation de la population
3. Sélection des meilleurs chromosomes qui répondent le mieux à la fonction sélective :
   - Sélection naturelle qui produit une nouvelle population.
4. Application des opérateurs génétiques :
   - Création d'une nouvelle population après application de la recombinaison et de la mutation.
5. Vérification du critère d'arrêt :
   - Si les critères d'arrêt sont atteints, fin du procédé, sinon retour à l'étape d'évaluation.

Fin

2.6 Vocabulaire des Algorithmes Génétiques

Du au fait que les AG s'appuient à la fois sur la génétique naturelle et l'intelligence artificielle, les terminologies utilisées pour les AG dans la littérature sont un mélange du naturel et de l'artificiel. Les termes usuels appliqués aux AG sont résumés dans le tableau ci-dessous

Tableau 2-1 Vocabulaire des l'AG

Algorithmes Génétiques	**Significations**
Chromosome (individu)	Solution
Gènes (variable)	Partie de la solution
Lieu	Position du gène
Allèles	Valeur du gène
Phénotype	Solution décodée
Génotype	Solution encodée

2.7 Création de la population

Lors de la genèse de la population, l'ensemble dans lequel est effectué la recherche de solutions doit être de dimension finie. La qualité et le choix du codage des chromosomes de la population est déterminant et conditionne le succès de la recherche de la meilleure solution. Le choix de la taille de la population est aussi important que le codage, car une population trop grande entraînera une recherche exhaustive, tandis qu'une population trop petite ne permettra pas à l'algorithme d'atteindre une solution optimale. Au cours des 10 dernières années, plusieurs méthodes de codage ont été créées pour la résolution des problèmes particuliers afin de fournir une facilité d'implémentation aux algorithmes génétiques [31]. Les méthodes de codage peuvent être classifiées comme suit :

- Le codage binaire qui consiste en des codes de [0,1]
- Le codage réel qui consiste en des nombres réels

- Le codage d'entier qui consiste en des nombres entiers

2.8 Évaluation de la population

Il s'agit d'un processus au cours duquel, les chromosomes de la population sont évalués par la fonction sélective (*fitness function*). C'est une fonction qui permet une évaluation de la population selon le critère de performance choisi. Elle permet ainsi de choisir les candidats les plus performants. Le critère de performance est fonction de la nature du problème. Dans certains cas la fonction sélective peut être établie comme une fonction de l'erreur. La plupart des algorithmes ont pour action de maximiser la valeur de la fonction sélective

2.9 Sélection

Le principe de sélection sur lequel s'appuient les Algorithmes génétiques est essentiellement la sélection naturelle darwinienne. La sélection des individus devant formés la nouvelle population, est un exercice d'équilibre, car un très grand niveau de sélection entraînera une fin prématurée de la recherche, tandis qu'à l'opposé un trop petit niveau de sélection, provoquera une recherche de la solution plus lente que nécessaire. Cependant, il est recommandé qu'au début du processus, un niveau de sélection peu élevé soit utilisé à cause du vaste espace de recherche et qu'à la fin du processus, un degré plus élevé de sélection soit utilisé afin de réduire l'espace de recherche. La sélection oriente le processus vers les zones de l'espace de recherche où sont susceptibles de se trouver les solutions au problème. Durant les deux dernières décennies, plusieurs méthodes de sélection ont été proposées, examinées et comparées. Dans la suite, la performance d'in individu (chromosome) est proportionnelle à l'évaluation de sa fonction sélective. Les types de

sélection les plus usuels sont : par la roulette, par tournoi, par rang, par l'élitisme et des compositions hybrides.

2.9.1 La sélection avec la roulette (wheel roulette selection)

Cette méthode fut proposée par John Holland [22], elle est reconnue comme la meilleure des techniques de sélection [31]. Ce mode de sélection s'appuie sur l'idée fondamentale suivante : déterminer la probabilité de sélection de chaque individu, proportionnellement à sa valeur de fonction sélective déterminée. Un modèle de roulette peut ensuite être mis en œuvre afin de déterminer les probabilités de sélection de chacun des chromosomes. La procédure de sélection est basée sur la roulette, qui est tournée un nombre de fois égal à la taille de la population. À chaque tour de la roulette, un chromosome est sélectionné afin de créer la nouvelle population. Comme illustré à la figure 2.9 chaque secteur de la roulette représente le pourcentage de représentativité des chromosomes sélectionnés pour la suite de la réalisation de l'algorithme. Un exemple plus concret de la mise en œuvre de cette méthode de sélection est effectué à la section 3.3.3.

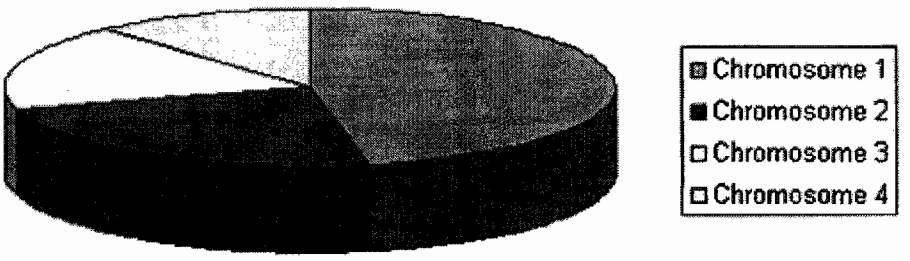

Figure 2.9 Sélection par la méthode de la roulette

2.9.2 La sélection par tournoi

Le principe de la sélection par tournoi [31] augmente les chances pour les individus de piètre qualité de participer à l'amélioration de la population. Une taille commune de tournoi

est 2. Cette méthode choisit aléatoirement deux individus de la population et les fait "combattre", celui qui a la fitness la plus élevé l'emporte et est sélectionné. On répète ce processus n fois de manière à obtenir les n individus qui serviront de parents.

2.9.3 La sélection par rang

Il s'agit d'une variante de la sélection avec la roulette. Dans cette méthode la roulette est également utilisée, cependant les secteurs de la roue ne sont plus proportionnels à la qualité des individus, mais à leur rang dans la population triée en fonction de la qualité des individus. Ainsi, il faut trier la population en fonction de la qualité des individus puis leur attribuer à chacun un rang. Les individus de moins bonne qualité obtiennent un rang faible (à partir de 1). En itérant sur chaque individu, on finit par attribuer le rang N au meilleur individu (où N est la taille de la population). La suite de la méthode consiste uniquement en la mise en oeuvre d'une roulette basée sur les rangs des individus. L'angle de chaque secteur de la roue sera proportionnel au rang de l'individu qu'il représente.

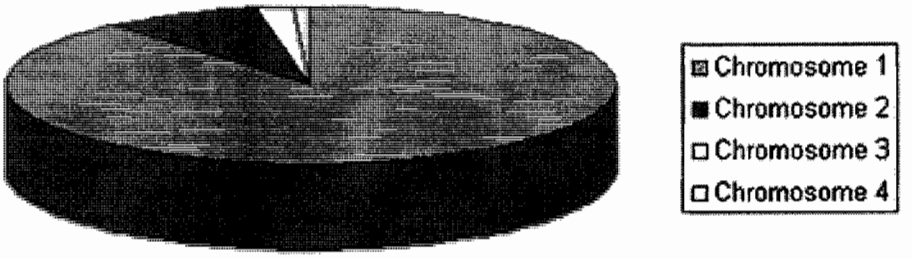

Figure 2.10 Sélection par la méthode du rang

2.9.4 L'élitisme

Cette méthode de sélection permet de mettre en avant les meilleurs individus de la population. Ce sont donc les individus les plus performants qui vont participer à

l'amélioration de la population. Cette méthode permet une convergence plus rapide des solutions, au détriment de la diversité des individus. Les individus pas très performants sont ainsi écartés, alors qu'ils auraient pu apporter le matériel nécessaire afin de créer de très bonnes solutions pour les générations suivantes.

En définitive, les méthodes de sélection permettent de déterminer quels individus nous allons croiser. Reste maintenant à définir comment nous allons les croiser à l'aide des opérateurs génétiques.

2.10 Les opérateurs génétiques

Les opérateurs génétiques effectuent des modifications aléatoires et ne peuvent garantir d'atteindre les enfants les mieux adaptés. Dans l'algorithme génétique conventionnel, la recombinaison est employée en tant qu'opérateur principal et l'exécution d'un système génétique dépend fortement d'elle. L'opérateur de mutation qui réalise des altérations aléatoires spontanées sur certains chromosomes, il est employé en tant qu'opérateur de fond. En fonction de la nature et de la complexité du problème la taille de la population peut être maintenue constante ou modifiée grâce aux opérateurs génétiques.

2.10.1 Recombinaison

La recombinaison génère de nouveaux individus en combinant l'information contenue dans les parents. Selon la représentation des variables, diverses méthodes peuvent être employées. On a ainsi principalement : les méthodes pour des variables à valeurs réelles et les méthodes pour des variables binaires.

2.10.1.1 Recombinaison arithmétique (valeurs réelles)

Le croisement arithmétique est propre à la représentation réelle. Il s'applique à une paire de chromosomes de la population et se résume à une moyenne pondérée des variables des deux parents. Soient {ai, bi, ci} et {aj, bj, cj} deux parents, et p un poids appartenant à l'intervalle [0, 1]. Les enfants issus des parents nommés plus haut sont décrits comme suit : {pai + (1-p)aj, pbi + (1-p)bj, pci + (1-p)cj}. Si nous considérons que p est un pourcentage [0, 100%], et que i et j sont nos deux parents, alors l'enfant i est constitué à p% du parent i et à (100-p)% du parent j, et réciproquement pour l'enfant j.

2.10.1.2 Recombinaison binaire

Cette section décrit les méthodes de recombinaison pour des individus constitués de variables binaires. Pendant la recombinaison de variables binaires, des portions de chromosomes sont échangées entre les chromosomes afin de former de nouveaux individus. L'endroit où se fait le partage d'information entre les chromosomes est appelé point de coupure. Le nombre de points de coupure distingue les méthodes. On a donc :

- La recombinaison binaire en un point

  Dans ce type de recombinaison, un point de croisement est choisi au hasard. À partir de ce point de coupure, les données sont échangées entre les individus afin de créer des enfants (Figure 2.11).

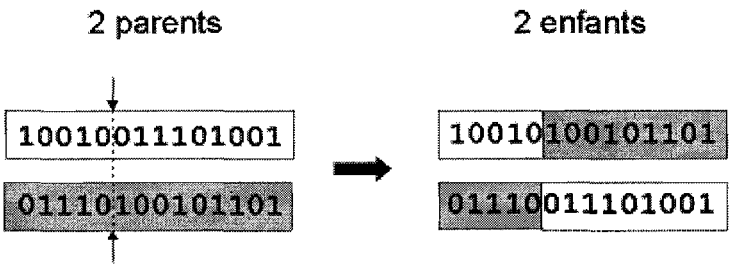

Figure 2.11 Recombinaison binaire en un point

- La recombinaison binaire en deux points

Dans ce cas, on choisit au hasard deux points de coupure à partir desquels on procède à l'échange de données, afin de créer des enfants comme montré à la figure ci-dessous :

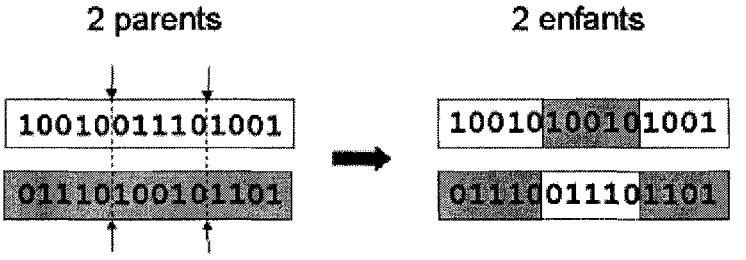

Figure 2.12 Recombinaison binaire en deux points

D'autres formes de recombinaison existent, de la recombinaison avec plusieurs points de coupure jusqu'au cas limite de la recombinaison uniforme assimilable à une recombinaison multipoints dont le nombre de points de coupure est inconnus. La mise en oeuvre de ce procédé est fort simple, elle consiste à définir de manière aléatoire un "masque", c'est-à-dire une chaîne de bits de même longueur que les chromosomes des parents sur lesquels il sera appliqué. Ce masque est destiné à savoir, pour chaque locus (position sur le chromosome), de quel parent le premier fils devra hériter du gène s'y trouvant. Si face à un locus le masque présente un 0, le fils héritera le gène s'y trouvant du parent n° 1, si il présente un 1

il en héritera du parent n° 2. La création du fils n° 2 se fait de manière symétrique : si pour un gène donné le masque indique que le fils n° 1 devra recevoir celui-ci du parent n° 1 alors le fils n° 2 le recevra du parent n°2, et si le fils n° 1 le reçoit du parent n° 2 alors le fils 2 le recevra du parent n° 1.

2.10.2 Mutation

Durant la mutation, des individus sont altérés de façon aléatoire. Ces altérations sont la plupart du temps petites. Selon la représentation des variables, différentes méthodes de mutation peuvent être employées. On a ainsi les méthodes pour des variables à valeurs réelles et la méthode pour des variables binaires.

2.10.2.1 Mutation à valeurs réelles

Ici la mutation est réalisée en rajoutant aux individus des réels de très petites valeurs crées de manière aléatoire.

2.10.2.2 Mutation binaire

La mutation est définie ici comme étant l'inversion d'un bit dans un chromosome (Figure 2.13). Le choix du bit à inverser est fait de manière aléatoire. Les mutations empêchent la recherche de la solution de stagner, en apportant de légère variation. Elles permettent d'assurer une recherche globale plutôt que local.

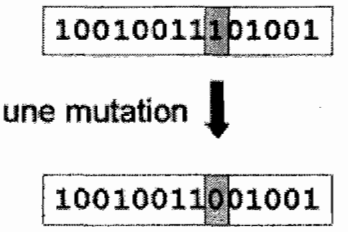

Figure 2.13 Mutation binaire

2.11 Avantages et inconvénients des algorithmes génétiques

2.11.1 Inconvénients des algorithmes génétiques

Les principales difficultés de mise en œuvre de la méthode se posent au niveau du réglage des processus de mutation et recombinaison, du choix de codage, de la taille de la population et de la dominance.

- Les opérateurs génétiques

  Les opérateurs de recombinaison et mutation ne tiennent pas compte de la structure du problème. Il s'agit de processus qui sont réalisés en suivant une logique bien établie peu importe la nature du problème. Ils doivent donc être adaptés et choisis en fonction du problème à résoudre.

- Codage

  Il est difficile de trouver un bon codage adapté à la structure d'un problème. Car chaque type de codage présente des forces et faiblesses dont il faut tenir compte. En fonction de la nature du problème un type codage peut être mieux adapté qu'un autre, ainsi dans les applications de recherche de maxima, le codage avec des réels est plus utilisé que le codage binaire.

- Taille de population

Comme mentionné plus haut, le choix de la taille de population est un exercice périlleux qui peut aboutir à la convergence ou non de la méthode vers la solution optimale. En d'autre terme : une taille de population mal adaptée fera traînée ou retardée la convergence vers la solution optimale, ou encore ne parviendra pas à trouver une solution convenable.

- La dominance

On peut arriver à une domination écrasante d'un individu "localement supérieur". Ceci entraînant une grave perte de diversité. Imaginons par exemple qu'on ait un individu ayant une fitness très élevée par rapport au reste de la population, disons dix fois supérieure, il n'est pas impossible qu'après quelques générations successives on se retrouve avec une population ne contenant que des copies de cet individu. Le problème est que cet individu avait une fitness très élevée, mais que cette fitness était toute relative, elle était très élevée mais seulement en comparaison des autres individus. On se retrouve donc face à problème connu sous le nom de "convergence prématurée; l'évolution se met donc à stagner et on atteindra alors jamais l'optimum, on restera bloqué sur un optimum local.

2.11.2 Avantages des algorithmes génétiques

Les trois principaux avantages résultant de l'application des algorithmes génétiques sont [31] :

1. Les AG ne nécessitent pas beaucoup d'outils mathématiques dans le traitement des problèmes d'optimisation. En raison de leur nature évolutionnaire, les AG rechercheront des solutions sans souci des particularités spécifiques à chaque

problème. Les AG peuvent résoudre n'importe quels types de problèmes d'optimisation, peu importe les contraintes (linéaire, non linéaire), définies sur des espaces discrets, continues ou sur des espaces de recherche, mixtes.

2. L'ergodicité des opérateurs évolutionnaires, rend les AG très performants dans la recherche globale (en probabilité). Les approches traditionnelles effectuent la recherche locale par un procédé de convergence par étape, qui compare les valeurs proches et tendent vers les valeurs optimales relatives.

3. L'AG appliqué avec un codage binaire prends en compte le phénomène de parallélisme implicite. Ce parallélisme tient compte de l'effet de quantification (virgules fixes), ce qui n'est pas le cas pour les méthodes traditionnelles pour lesquelles une étude de quantification doit être effectuée si on désire travailler en virgule fixe.

Chapitre 3 - Identification d'un système ARMA

Ce chapitre présente le détail de la mise en œuvre d'un algorithme génétique pour l'identification de paramètres d'un système de type auto régressif à moyenne mobile ou plus communément appelé ARMA (*Auto Regressive Mean Average*). Le système de type ARMA sera défini à la section 3.2 tandis que les résultats ainsi que l'étude comparative vis-à-vis des méthodes du gradient stochastique seront présentés au chapitre 5.

3.1 Identification adaptative

La détermination d'un modèle mathématique pour un système inconnu à travers l'observation de ces données d'entrées et sorties est généralement connu comme l'identification d'un système. L'identification d'un système est réalisée par l'ajustement des paramètres pour un modèle donné, jusqu'à ce que la sortie pour séquence d'entrées particulières, coïncide autant que possible avec séquence des sorties mesurées du système identifié [32]. L'identification comprend les étapes suivantes : la planification expérimentale, la sélection d'un modèle de structure, l'estimation des paramètres et enfin la validation du modèle [10]. L'estimation des paramètres étant la principale tâche lors de l'identification d'un système [2], on s'attardera donc dans la suite uniquement sur l'estimation des paramètres.

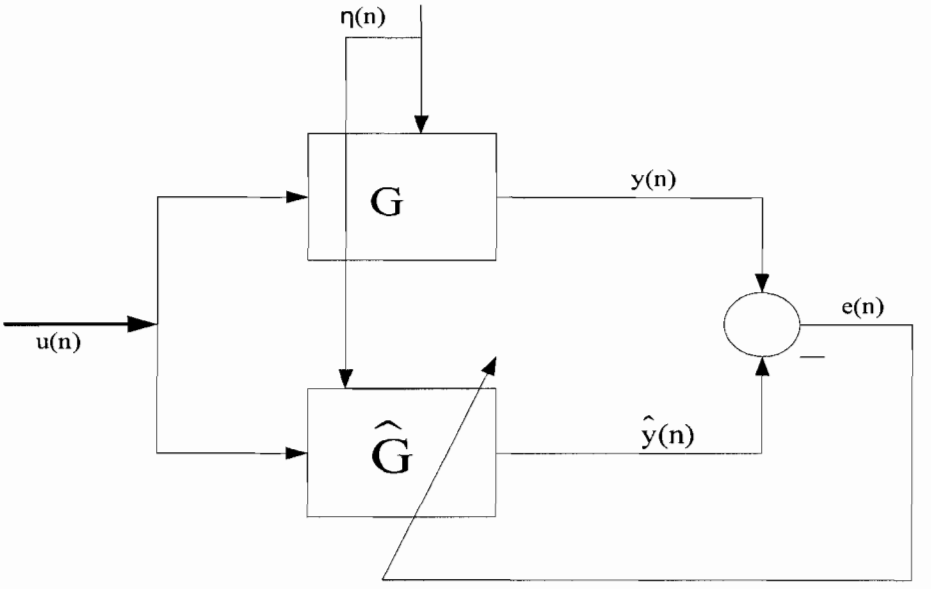

Figure 3.1 Identification d'un système linéaire

Le système à identifier est celui présenté à la figure 4.1. *G* représente le système idéal et $\widehat{G}$ un modèle du système estimé. L'identification du système se déroule comme suit : le signal d'entrée *u(n)* est appliqué à l'entrée du système (canal) à *M* coefficients avec *M* représentant le nombre de paramètres du canal. Un vecteur $w(n)$, ou vecteur des poids du filtre, de longueur identique aux nombres de paramètres du canal, contient l'information concernant les estimations des coefficients du canal. Le signal $\hat{y}(n)$ estimé est calculé en fonction des données *u(n)* présentées à l'entrée du canal et du vecteur des poids $w(n)$ selon l'équation :

$$\hat{y}(n) = \mathbf{w}(n)^T \mathbf{u}(n) \tag{3.1}$$

avec $\mathbf{w}(n) = [w_1(n), w_2(n), ..., w_M(n)]^T$ et

$$e(n) = y(n) - \hat{y}(n) \tag{3.2}$$

L'erreur d'adaptation (3.2) permet de faire l'adaptation des coefficients du filtre. L'objectif étant de minimiser l'erreur d'adaptation, de tel sorte que les poids estimés soient aussi proches que possible des coefficients réels du système.

3.2 Modèle ARMA

On appelle modèle ou processus ARMA d'ordre P et Q, un signal $y(n)$ résultant du filtrage d'un signal $x(n)$ par un filtre linéaire comportant P pôles et Q zéros. Si le filtre n'a que P pôles, il s'agit d'un processus autorégressif d'ordre P que l'on note AR (P). Si le filtre n'a que Q zéros, il s'agit d'un processus à moyenne mobile (MA – Mean Average) d'ordre Q que l'on note MA(Q). Dans le cas général, on parle d'un processus ARMA(P,Q). Le modèle ARMA peut être utilisé pour représenter tout processus aléatoire ayant une densité spectrale uniforme. De ce fait le modèle ARMA est largement utilisé dans de nombreuses applications [33].

3.2.1 Modèle ARMA linéaire

En s'appuyant sur le système présenté à la figure 4.1 et l'article de Duong-Stubberud [13], le modèle du système ARMA identifié dans le cas linéaire est décrit comme suit :

$$\begin{aligned} y(n+T_s) = {} & a_0 y(n) + a_1 y(n-T_s) + \cdots + a_M y(n-MT_s) \\ & b_0 u(n) + b_1 u(n-T_s) + \cdots + b_N u(n-NT_s) \\ & c_0 \eta(n) + c_1 \eta(n-T_s) + \cdots + c_L \eta(n-LT_s) \end{aligned} \tag{3.3}$$

Avec $y(n)$ représentant la sortie du système, T_s la période d'échantillonnage, $u(n)$ l'entrée du système et $\eta(n)$ un bruit blanc gaussien de moyenne nulle et de variance

unitaire. Pour ce système, nous considérons qu'il n'existe aucune corrélation entre $\eta(n)$ et la sortie. Les coefficients du système, peuvent être représentés comme suit : $\mathbf{w}(n) = [w_1(n), w_2(n), ..., w_M(n)]^T$ avec chacun des w représentants un coefficient du système.

Le but est d'arriver des coefficients estimés $\hat{\mathbf{w}}(n) = [\hat{w}_1(n), \hat{w}_2(n), ..., \hat{w}_M(n)]^T$ pour le système, de telle sorte que le critère de performance ε (3.6) soit minimisé.

En considérant les coefficients et les divers retards connus pour le système décrit à l'équation (3.3), le système à identifier est exprimé comme suit :

$$y_L(n) = 1.5y_L(n-1) - 0.7y_L(n-2) + u(n-5) + 0.5u(n-6) + \eta(n) - \eta(n-1) + 0.2\eta(n-2) \tag{3.4}$$

Avec le signal d'entrée $u(n) = (1.1)^{-n}$ et le bruit blanc Gaussien $\eta(n)$ de moyenne nulle et de variance unitaire. À partir du modèle du système idéal, le système estimé est décrit par l'équation qui suit:

$$\hat{y}_L(n) = a_1\hat{y}_L(n-1) - a_2\hat{y}_L(n-2) + b_1u(n-5) + b_2u(n-6) + c_1\eta(n) - c_2\eta(n-1) + c_3\eta(n-2) \tag{3.5}$$

La performance du système est mesurée avec le MSE relatif, comme suit :

$$\varepsilon(n) = \frac{\sum_{i=1}^{M}\left(w_i(n) - \hat{w}_i(n)\right)^2}{\sum_{i=1}^{M} w_i^2} \tag{3.6}$$

Avec M représentant la dimension du filtre. Comme décrit dans le cas général plus haut, l'objectif est de choisir les coefficients $[a_1\ a_2\ b_1\ b_2\ c_1\ c_2\ c_3]$, afin de minimiser ε .

3.2.2 Modèle ARMA non linéaire

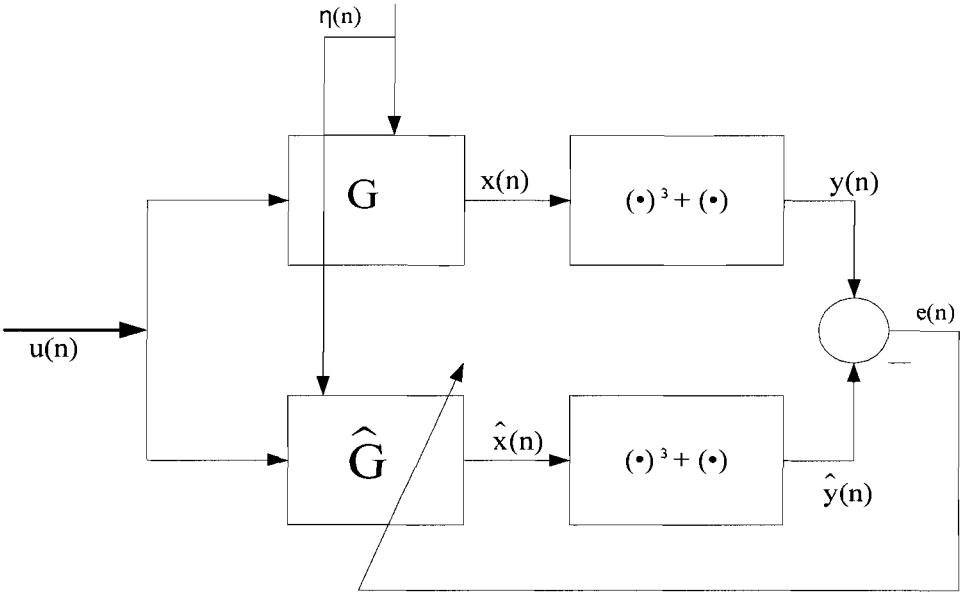

Figure 3.2 Identification d'un système non linéaire

L'identification du système non linéaire s'apparente à celle du système linéaire à la différence qu'une non linéarité est ajoutée au système (figure 3.2). Le système non linéaire est décrit par les équations de sorties désirées (3.7) et estimées (3.8) respectives :

$$y_L(n) = \left(y_L(n)\right)^3 + y_L(n) \tag{3.7}$$

$$\hat{y}_L(n) = \left(\hat{y}_L(n)\right)^3 + \hat{y}_L(n) \tag{3.8}$$

L'entrée $u(n)$ et le bruit blanc gaussien $\eta(n)$, sont les mêmes que dans le cas linéaire. Comme dans le cas du système linéaire, l'objectif est de choisir les coefficients $[a_1\ a_2\ b_1\ b_2\ c_1\ c_2\ c_3]$, afin de minimiser ε (3.6).

3.3 Algorithmes Génétiques pour l'identification d'un système ARMA

Dans cette section, le détail de l'application des AG à l'identification du système ARMA décrit à la section 3.2 est effectué. Les grandes étapes de la mise en œuvre des AG tels que décrites au chapitre 2 (figure 2.8) sont maintenues ici. La figure 3.3 est une version plus détaillée de l'algorithme génétique appliquée au présent travail. Les détails de chaque opération sont présentés dans cette section.

3.3.1 Création de la population

Il s'agit du début de la méthode des AG, qui commence par la création d'une population initiale. La population initiale, est une population aléatoire de chromosomes dont la taille varie selon la dimension du problème à résoudre. Cet intervalle est déterminé en se fiant au fait que si l'on suppose la représentation binaire de 7 coefficients (gènes) sur 8 bits, on se retrouve avec $2^{7\times8}$ possibilités de solution pour un chromosome. Au regard du nombre de possibilités de solutions possibles pour un chromosome le choix d'une taille de population compris dans l'intervalle [10, 40] semble judicieux, car une population trop grande entraînera une recherche exhaustive, tandis qu'une population trop petite ne permettra pas à l'algorithme d'atteindre une solution optimale vers un minimum globale.

Le codage choisi est de type réel afin de faciliter l'évaluation des chromosomes. Cependant à certaines étapes du processus les chromosomes seront convertis en données de types binaires. Chaque chromosome de la population est une concaténation de réel

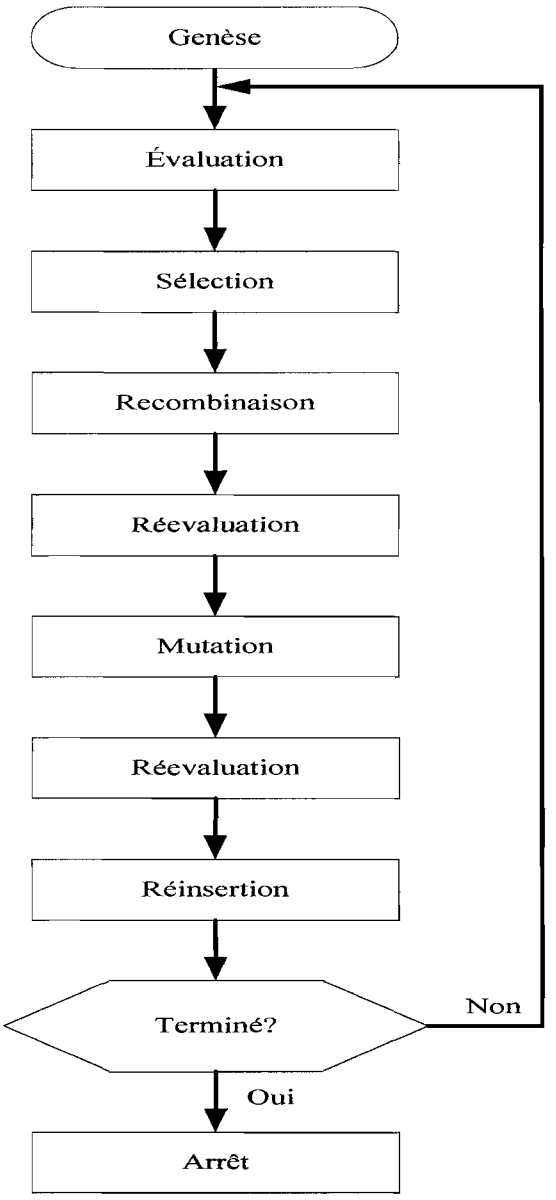

Figure 3.3 Structure générale d'un Algorithme Génétique appliqué à l'identification d'un système de type ARMA

correspondant au nombre de coefficients à identifier (voir figure 3.4). Chacun des réels ou gènes appartient à un intervalle créant un espace de recherche qui est plus grand que la valeur des coefficients que l'on souhaite estimer. Ceci permet une diversité au niveau des chromosomes et permet également à la recherche de ne pas stagner très rapidement.

a_1	a_2	b_1	b_2	c_1	c_2	c_3
2	4	1	8	6	1	7

Figure 3.4 Représentation d'un chromosome un processus aléatoire de gènes dans l'intervalle [0,10]

3.3.2 Évaluation de la population

La population est évaluée grâce à la fonction sélective. Cette dernière permet de déterminer les individus les plus performants. En s'appuyant sur l'article de W-Y. Wang et Y-H.Li [34], la fonction sélective est définie comme étant l'inverse de l'erreur. Selon (3.2) pour chaque chromosome j, nous avons :

$$f_j = \frac{1}{e_j} \tag{3.9}$$

L'erreur e du chromosome j, à la génération i est définie comme suit :

$$e_j^{\,i} = \frac{1}{W}\sum_{n=1}^{W}\left\|(y(n) - \hat{y}_j^{\,i}(n))\right\| \tag{3.10}$$

Avec W qui représente la fenêtre au travers de laquelle l'erreur est accumulée. Ce calcul de l'erreur au travers d'une fenêtre a pour objectif principal de réduire la complexité de l'AG. À chaque génération, les AG essaient de minimiser l'erreur. C'est à partir de cette erreur minimale qu'est déterminé le chromosome le plus performant. Suite à l'obtention de ce chromosome, la qualité de la méthode d'identification du système à chaque génération, est calculée selon l'équation (3.6).

3.3.3 Sélection

Après avoir effectué l'évaluation des chromosomes de la population, les individus qui répondent le mieux au critère de performance imposé par la fonction sélective vont être sélectionnés, tandis que ceux qui sont les moins bien adaptés vont être rejetés. La méthode de sélection utilisée ici est la méthode de la roulette (*wheel roulette*). Elle est employée dans de nombreuses applications [8] et reconnue pour la simplicité de sa mise en œuvre. Les étapes de la sélection avec la roulette se déroule comme suit :

a. Calcul du total de la fonction sélective pour la population de m chromosomes

$$F = \sum_{j=1}^{m} f_j \tag{3.11}$$

b. Calcul de la probabilité de sélection pour chacun des chromosomes

$$P_j = \frac{f_j}{F} \tag{3.12}$$

c. Calcul de la probabilité cumulative pour chacun des chromosomes

$$q_j = \sum_{j=1}^{m} P_j \tag{3.13}$$

d. Le processus de sélection proprement dit commence en tournant la roue m fois, à chaque tour de la roue un chromosome (*Pop*) est choisi au hasard afin de former une nouvelle population de la manière suivante :

   1. Génération d'un nombre réel aléatoire r, appartenant à l'intervalle [0, 1].

   2. Si $r \leq q_1$, alors le chromosome Pop_1 est choisi, sinon sélectionner le j^{ieme} chromosome $Pop_j (2 \leq j \leq m)$ de telle sorte que $q_{j-1} < r \leq q_j$.

À la fin de ce processus de sélection, une nouvelle population est obtenue, formée uniquement de chromosomes qui sont les mieux adaptés.

3.3.4 Recombinaison

Le type de recombinaison utilisé ici, est la recombinaison binaire en un point, qui permet un plus grand échange de matériel génétique entre les parents afin de donner de nouveaux chromosomes assez différents des parents qui permettront à la population de se renouvelée et à la recherche de ne pas stagner trop rapidement. Avant de procéder à l'échange de gènes entre parents, il faudrait déterminer les d'individus de la population qui vont participer à la recombinaison.

3.3.4.1 Choix des chromosomes à recombiner

Le choix des chromosomes pouvant participer à la recombinaison se déroule de la manière suivante :

1. Déterminer le pourcentage p_c d'individus de la population pouvant participer à la recombinaison :

   Le pourcentage d'individus devant être recombiné dépend de l'application, de la taille de la population et il est compris entre 0 et 100%. Le choix du pourcentage de la population devant participer aux opérations génétiques de recombinaison et de mutation est déterminant pour la convergence de l'algorithme.

2. Le choix des chromosomes à recombiner est réalisé suivant la procédure suivante :

   début
   Tant que $j \leq m$
   générer un nombre aléatoire r_j *comprit dans l'intervalle [0, 1]*

si $r_j < p_c$ *alors*

Choisir Pop_j *comme parent pour la recombinaison*

sinon

Faire $j = j + 1$

fin

Après avoir sélectionné les individus devant participer à la recombinaison, la prochaine étape consiste à convertir nos individus qui sont jusqu'à lors des concaténations d'entiers en concaténations de valeur binaires.

3.3.4.2 Conversion de nombre réel en binaire

Dans l'application des AG, la conversion de réel à binaire est un exercice facile à réaliser, cependant lorsqu'il s'agit de convertir des réels à valeurs négatives, l'exercice devient plus complexe. L'outil de simulation (Matlab) avec lequel le travail présent a été effectué n'est pas capable de convertir des réels négatifs en représentation binaire, une astuce basée sur [4] permettant de contourner cette limitation de l'outil de simulation est présentée comme suit :

Considérons le gène a_1, compris dans l'intervalle $[a_{1\min}, a_{1\max}]$. Considérons également b, le nombre de bit sur lequel est représenté a_1. La représentation de a_1 selon la formulation :

$$a_1 = \frac{(a_1 - a_{1\min})}{((a_{1\max} - a_{1\max})/(2^b - 1))} \tag{3.14}$$

permet d'avoir toujours a_1 positif et ainsi donc de pouvoir utiliser la fonction Matlab *dec2bin*, afin de procéder à la conversion binaire de a_1 que nous nommerons a_{1bin}. Pour retrouver la véritable valeur réelle de a_1 il suffit d'appliquer la formule qui suit :

$$a_1 = a_{1\min} + (decimal(a_{1bin})) \times \frac{(a_{1\max} - a_{1\min})}{(2^b - 1)} \tag{3.15}$$

avec $decimal(a_{1bin})$ qui correspond à la valeur réelle de a_{1bin} fournit par la fonction Matlab *bin2dec*.

Une fois que le choix des chromosomes et la conversion de ces derniers en valeur binaire ont été effectués, la suite du processus est le choix d'un point de coupure à partir duquel l'échange de patrimoine génétique entre parents sera effectué selon la figure 2.3. Le point de coupure est un entier créé de façon aléatoire. Il est compris entre 1 et la taille maximale d'un chromosome. La dernière étape consiste donc finalement à la recombinaison proprement dite.

3.3.5 Réévaluation de la population

Contrairement à la méthode des AG classique, la population issue de la recombinaison est évaluée de nouveau afin de fournir un nouveau calcul d'erreur. Cette nouvelle évaluation qui est calquée sur celle effectuée à la section 3.3.2 permet d'identifier de nouveau juste après les opérateurs, les individus les mieux adaptés et ceux qui le sont moins. Toutefois avant de procéder à la nouvelle évaluation, les enfants obtenus grâce à la recombinaison sont convertis en valeur réelle selon. 3.16.

3.3.6 Mutation

Le type de mutation utilisé ici, est la mutation binaire selon la section 2.6.2. Le choix du bit à inverser est fait de manière aléatoire. L'intervalle dans lequel est réalisé le choix du bit à altérer est compris entre 1 et la taille maximale d'un chromosome. La mise en œuvre de la mutation est régie par les mêmes étapes que celles de la recombinaison, à savoir : le choix des chromosomes à muter selon un pourcentage p_m qui est compris dans l'intervalle entre 0 et 100%, la conversion des chromosomes de valeurs réelles à binaires, puis finalement la réalisation de la mutation proprement dite. Les étapes de choix des chromosomes à muter et de conversion sont *idem* à celles effectuées lors de la recombinaison.

3.3.7 Réévaluation et réinsertion de la population

À ce niveau les chromosomes issus de la mutation sont évalués comme à la section 3.3.5. Grâce à cette nouvelle évaluation, les chromosomes sont répertoriés en fonction de leurs performances. Une fois la réévaluation terminée, les chromosomes les moins performants issus de la mutation sont remplacés par les chromosomes les plus performants issus de la recombinaison.

La nouvelle population ainsi obtenue constitue la prochaine génération d'individus, qui sera réinsérée dans le processus à son tout début au niveau de la première évaluation.

3.3.8 Critère d'arrêt

Tant que le critère d'arrêt n'est pas atteint, le processus continuera jusqu'à ce qu'il soit atteint. Ici le critère d'arrêt est un nombre de générations fixé qui doit être atteint.

Le diagramme ci-dessous présente un résumé de la méthode des AG appliquée à l'identification d'un système de type ARMA.

Chapitre 4 - Résultats d'identification

Les méthodes d'identifications classiques que sont le LMS, le NLMS et le RLS ont été présentées à travers leurs différentes équations de mise en œuvre au chapitre 2. Les équations décrites dans le chapitre 2 sont celles ayant servi à l'identification du système ARMA. La méthode s'appuyant sur les algorithmes génétiques a été décrite en détail dans le chapitre 4. Dans la suite de ce chapitre, la mise en œuvre de la méthode à travers le choix des paramètres des AG sera présentée. Une étude comparative entre les méthodes classiques et les AG sera présentée. La comparaison des résultats sera également présentée dans des tableaux où les valeurs réelles des paramètres et celles obtenues avec les algorithmes seront observées. La dernière partie de l'étude comparative sera axée sur une étude de complexité qui démontrera que les AG sont compétitifs par rapport aux méthodes classiques de références.

4.1 Choix des paramètres des méthodes classiques

Les paramètres dont il s'agit ici sont : le pas de convergence (μ) pour le LMS et le NLMS, le facteur d'oubli (λ) et la constante (β) d'initialisation de la matrice de covariance pour le RLS. Ces paramètres ont été déterminés afin de maximiser les performances des méthodes classiques pour l'identification de notre système (système linéaire et non linéaire) décrit dans le chapitre précédent. Cette performance est déterminée par les principaux critères suivants : vitesse de convergence et erreur asymptotique. Le tableau 4.1 présente

les valeurs des paramètres d'adaptation des méthodes dans diverses conditions de simulation.

Tableau 4.1 Paramètres d'adaptation du LMS, NLMS et RLS

Méthodes	μ	λ	β
Virgule flottante, 24 bits, 16 bits, 12 bits			
LMS	0.075		
NLMS	1.4		
RLS		1	5×10^{-3}
Virgule fixe, 8 bits, 6 bits			
LMS	0.1		
NLMS	1.3		
RLS		0.93	5×10^{-3}

Grâce au tableau 4.1, on constate déjà la dépendance des méthodes classiques aux conditions du nombre de bits de quantification. Les paramètres doivent être ajustés afin de fournir des résultats optimaux. À savoir, la minimisation de l'erreur asymptotique d'identification des paramètres du système selon (3.6).

4.2 Évaluation des paramètres de l'AG

Le nombre de paramètres à fixer pour les AG est plus important que ceux des méthodes classiques. En effet, comme mentionné à la section 2.11, la performance des AG dépend des paramètres suivants : la taille de la population, les opérateurs génétiques avec les

pourcentages de population sur lesquelles ils seront appliqué, le codage, ainsi que le mode de sélection. On s'attardera ici sur les paramètres vitaux qui influencent grandement la méthode que sont la population et les opérateurs génétiques.

4.2.1 Taille de la population

Le choix de la taille de la population est une étape cruciale dans la mise en œuvre d'un AG, car une population trop grande entraînera une recherche exhaustive, tandis qu'une population trop petite ne permettra pas à l'algorithme d'atteindre une solution optimale. En s'appuyant sur une taille de population d'individus entiers dont la taille varie dans l'intervalle [10,40]. Des simulations pour les systèmes ARMA linéaire (figure 3.1) et non linéaire (figure 3.2) ont été effectuées afin d'évaluer l'influence de la taille de la population sur la performance du système. Les courbes des figures 4.1 et 4.2 sont réalisées avec un niveau de bruit élevé (SNR=0 dB) pour le signal $\eta(n)$, elles représentent une moyenne des courbes de convergence de 50 réalisations de simulations, la recombinaison et la mutation sont effectuées sur l'ensemble de la population tandis le nombre de générations est 100. Pour ces simulations, la recombinaison et la mutation sont effectuées sur l'ensemble de la population.

Les simulations effectuées pour plusieurs tailles de population dans les cas linéaire et non linéaire du système ARMA, montrent très clairement qu'une population de dix individus est suffisante et donne de meilleurs résultats qu'une plus grande population pour l'identification du canal ARMA. On observe également une remontée des courbes d'erreurs autour de la 5^{eme} génération qui est due à la fenêtre d'accumulation de l'erreur ($W=1$) qui fournit une erreur non pondérée sur plusieurs accumulations.

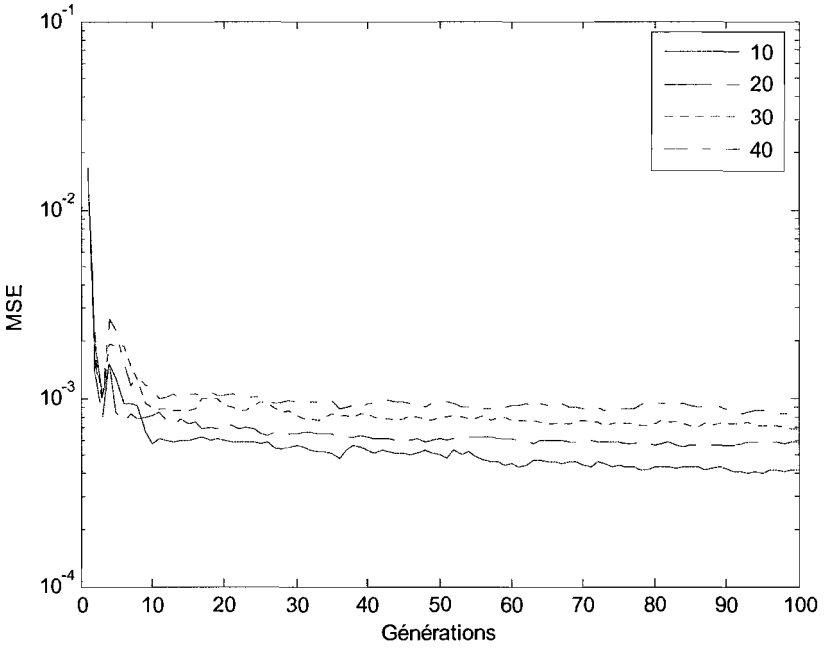

Figure 4.1 Application des AG à l'identification d'un canal linéaire ARMA pour diverses tailles de population

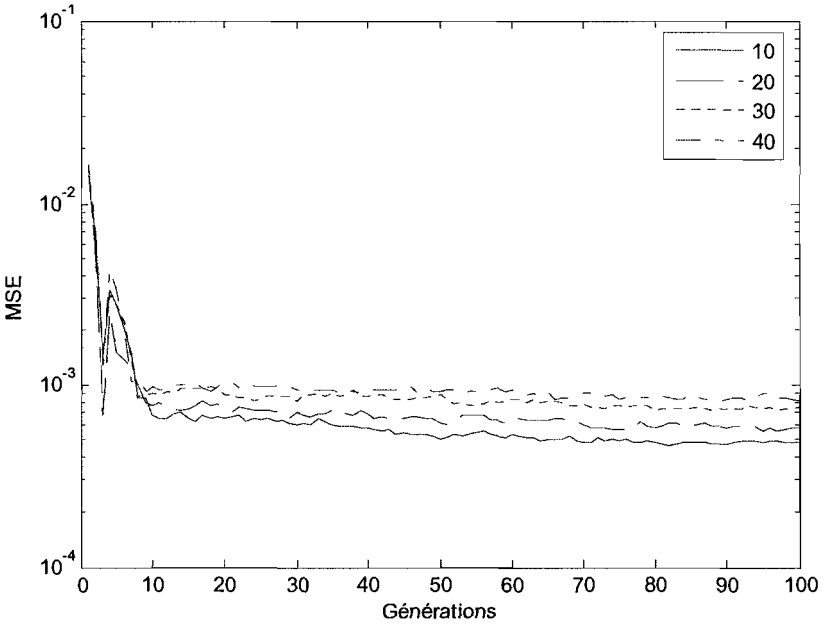

Figure 4.2 Application des AG à l'identification d'un canal non linéaire ARMA pour diverses tailles de population

4.2.2 Recombinaison et mutation

Hormis l'importance du choix du type de recombinaison et de mutation, le pourcentage d'individus à être recombinés et mutés l'est tout autant, car le bon choix du pourcentage des individus pouvant participer aux opérations génétiques est capital pour une bonne convergence de la méthode. Des simulations pour les systèmes ARMA linéaire et non linéaire ont été effectuées afin de déterminer le pourcentage d'individus pour les deux opérateurs génétiques les plus aptes à donner les meilleurs résultats. Dans le cas du canal linéaire, les courbes des figures 4.3 et 4.4 ont été réalisées avec un niveau de bruit SNR=0dB pour le signal $\eta(n)$, elles représentent une moyenne de 50 réalisations des simulations, avec une population de 10 individus et un nombre de générations de 100.

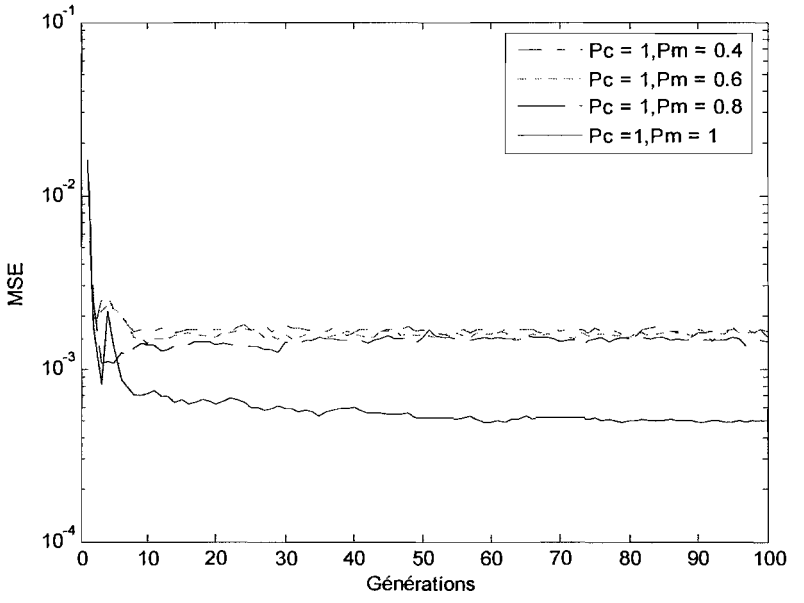

Figure 4.3 Application des AG à l'identification d'un canal linéaire ARMA pour divers pourcentage de mutation (p_m) et de recombinaison (p_c)

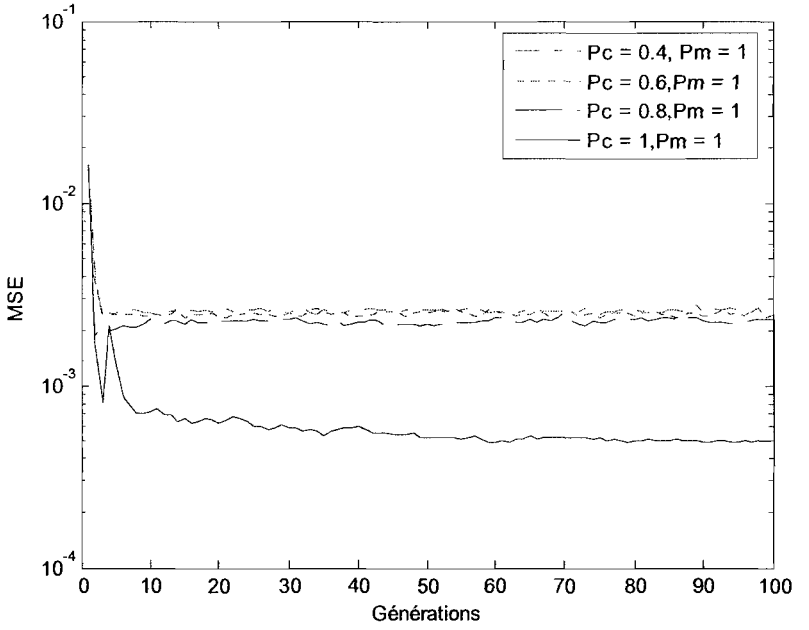

Figure 4.4 Application des AG à l'identification d'un canal linéaire ARMA pour divers pourcentage de mutation (p_m) et de recombinaison (p_c)

Les courbes des figures 5.3 et 5.4 permettent d'observer que pour obtenir les meilleurs résultats (MSE) et empêcher que l'algorithme stagne très rapidement les opérateurs génétiques doivent être appliqué à l'ensemble de la population (p_m =100%, p_c =100%).

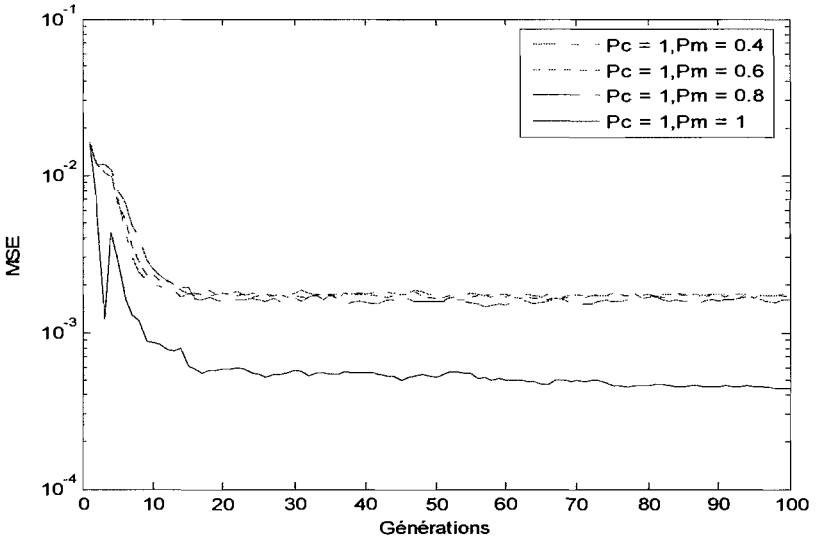

Figure 4.5 Application des AG à l'identification d'un canal non linéaire ARMA pour divers pourcentage de mutation (p_m) et de recombinaison (p_c)

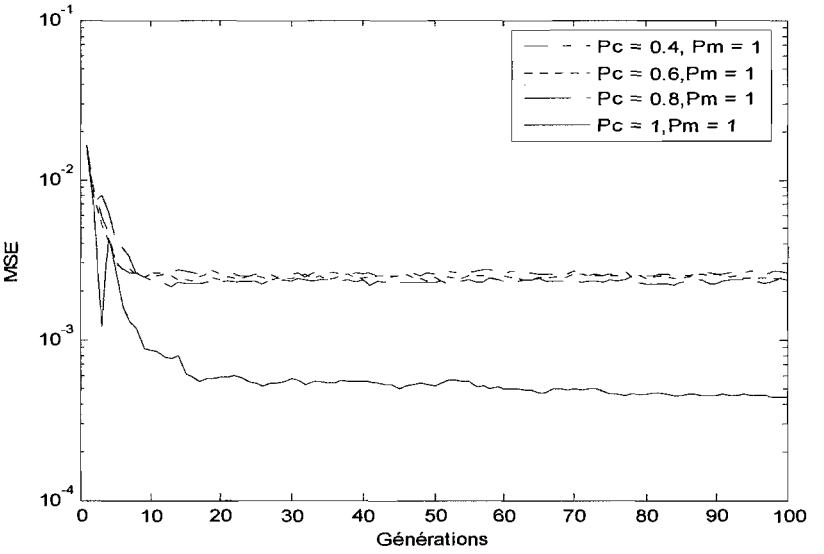

Figure 4.6 Application des AG à l'identification d'un canal non linéaire ARMA pour divers pourcentage de mutation (p_m) et de recombinaison (p_c)

Les conditions de simulation pour le canal non linéaire sont les mêmes que celles du canal linéaire décrit plus haut. Comme dans le cas du canal linéaire, les figures 4.5 et 4.6 démontrent que les meilleurs résultats (MSE) sont obtenus lorsque les opérateurs génétiques sont appliqués à l'ensemble de la population (p_m =100%, p_c =100%).

4.2.3 Taille de la fenêtre de calcul

Lors du calcul de l'erreur, une fenêtre (W) d'accumulation des erreurs est utilisée. Ce calcul de l'erreur cumulée au travers d'une fenêtre d'observation permet d'assurer une meilleure convergence de l'AG.

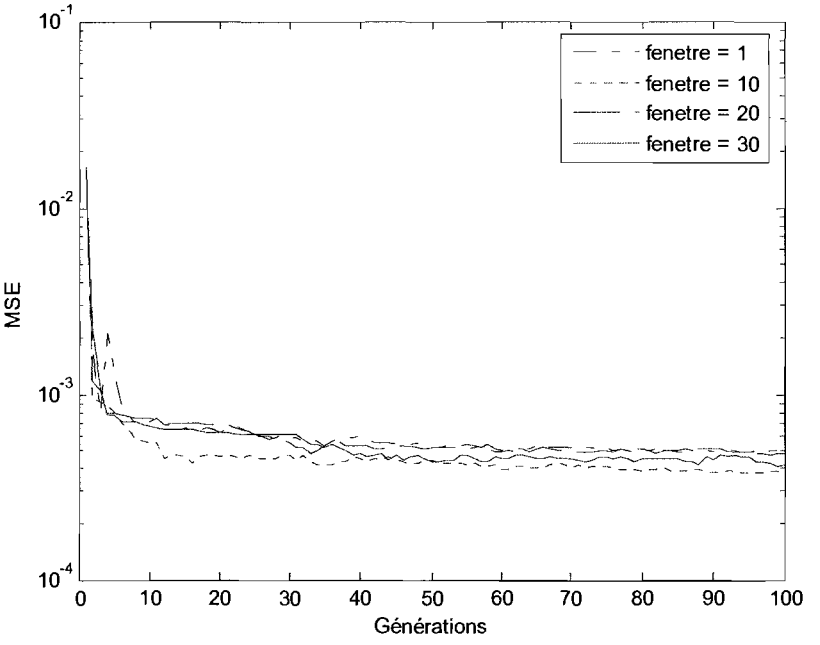

Figure 4.7 Application des AG à l'identification d'un canal linéaire ARMA pour diverses tailles de fenêtre

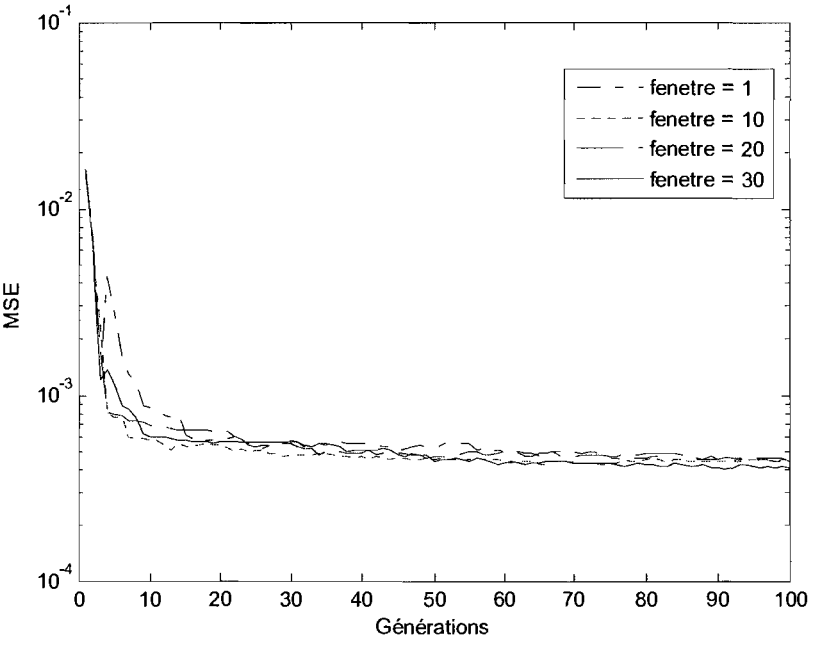

Figure 4.8 Application des AG à l'identification d'un canal non linéaire ARMA pour diverses tailles de fenêtre

Au regard des figures 4.7 et 4.8 nous constatons que la taille de la fenêtre influence peu la convergence de la méthode. Nous constatons également que la fenêtre W=10 obtient le minimum d'erreur le plus faible après moins de générations. On observe également une

remontée de la courbe d'erreur avec une fenêtre de 1. Cette remontée est du au fait que l'erreur pour une fenêtre de 1 n'est pas une moyenne des accumulations. Afin de réduire la complexité des calculs et de s'aligner par rapport aux autres méthodes étudiées dans ce travail et faire une étude comparative juste, la taille de la fenêtre choisie est 1.

4.3 Procédure de la comparaison

Les courbes et tableaux présentés dans la suite seront ceux de l'erreur quadratique moyenne (MSE) en fonction du nombre d'itérations ou générations. Le nombre d'itérations présentées ici a été normalisé afin de faire une étude plus juste et objective entre les AG et les autres méthodes. La normalisation consiste à multiplier le nombre de générations effectuées par les AG par le nombre d'individus de la population, car chacun des individus de la population représente un filtre d'équivalence ou FIR-LMS. Dans notre cas la multiplication est de 10 compte tenu de la taille de la population de 10 individus. Les courbes sont obtenues en faisant la moyenne de 50 réalisations de simulations, pour 500 données transmises avec un niveau de bruit SNR=0dB pour le signal $\eta(n)$ ayant une variance unitaire afin de rendre le canal difficile à identifier.

4.4 Étude comparative entre les méthodes classiques et les AG

4.4.1 Résultats de simulation pour le canal linéaire

Les figures 4.9 à 4.14 présentent les résultats de simulation du canal linéaire, identifié à l'aide du LMS, du NLMS, du RLS et des AG. Les figures sont réalisées pour diverses valeurs de bits, mais également en virgule flottante afin de montrer la robustesse des AG et surtout leur capacité à opérer avec très peu de bits.

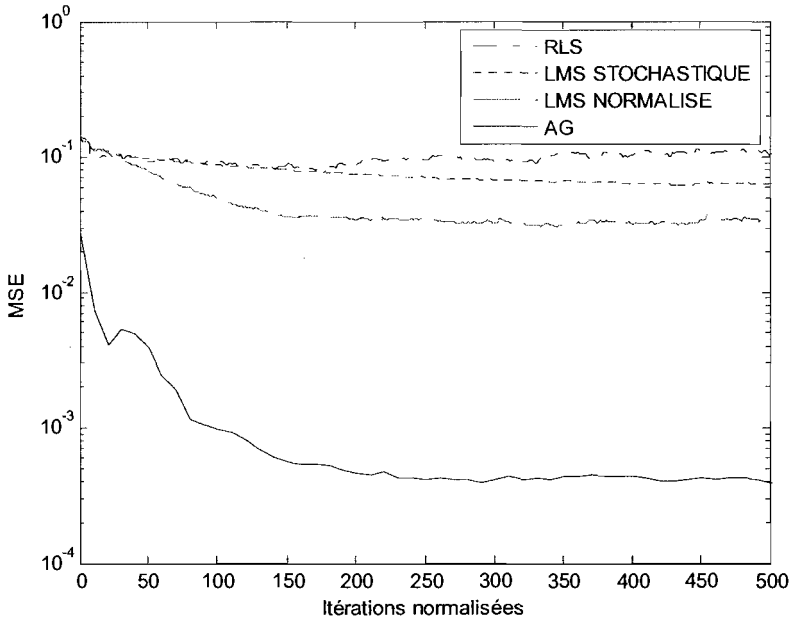

Figure 4.9 Identification du canal linéaire ARMA par LMS, NLMS, RLS et AG pour 6 bits

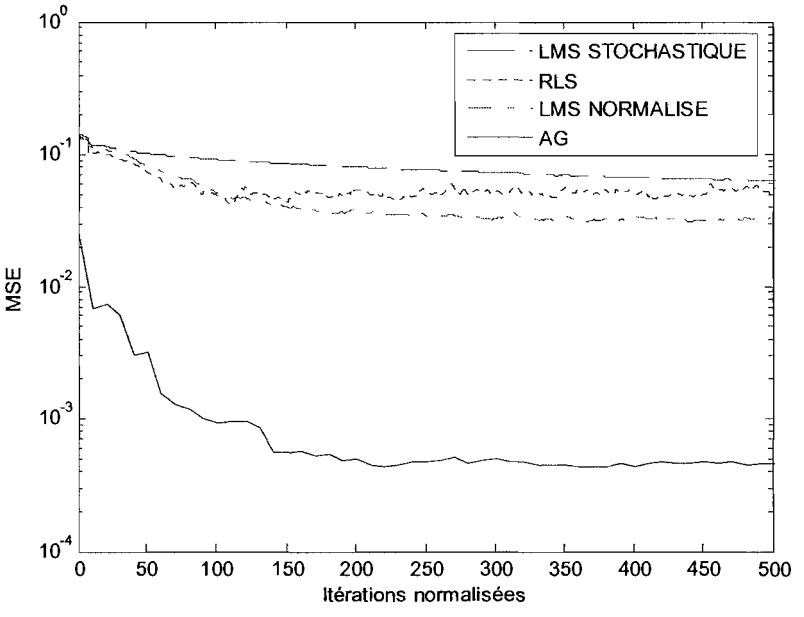

Figure 4.10 Identification du canal linéaire ARMA par LMS, NLMS, RLS et AG pour 8 bits

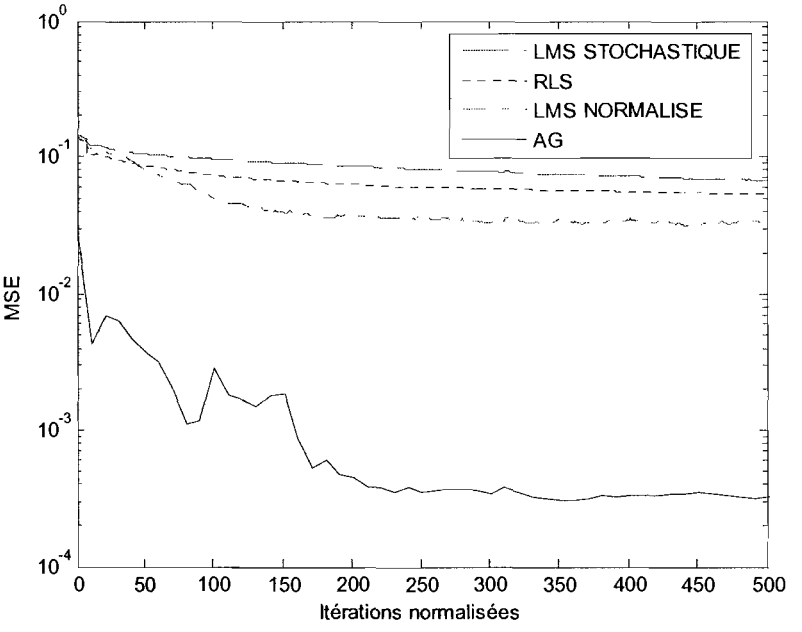

Figure 4.11 Identification du canal linéaire ARMA par LMS, NLMS, RLS et AG pour 12 bits

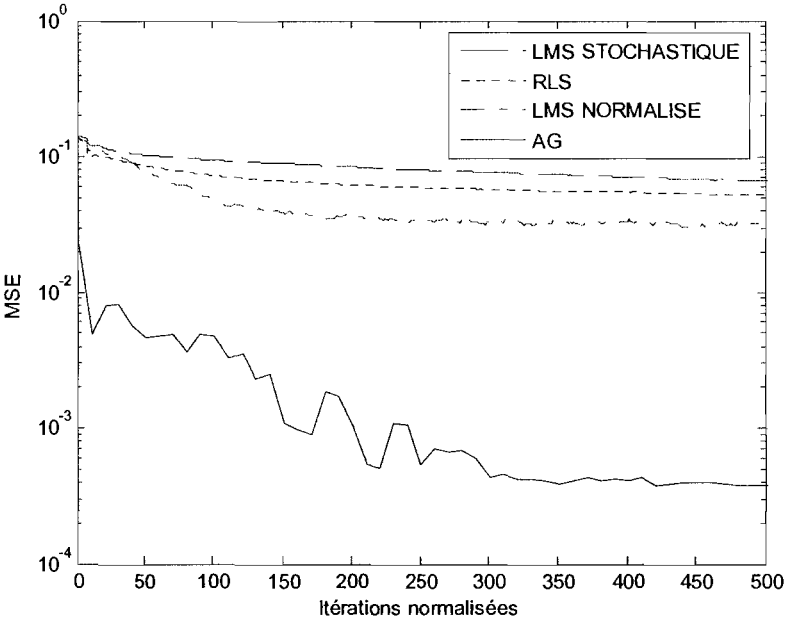

Figure 4.12 Identification du canal linéaire ARMA par LMS, NLMS, RLS et AG pour 16 bits

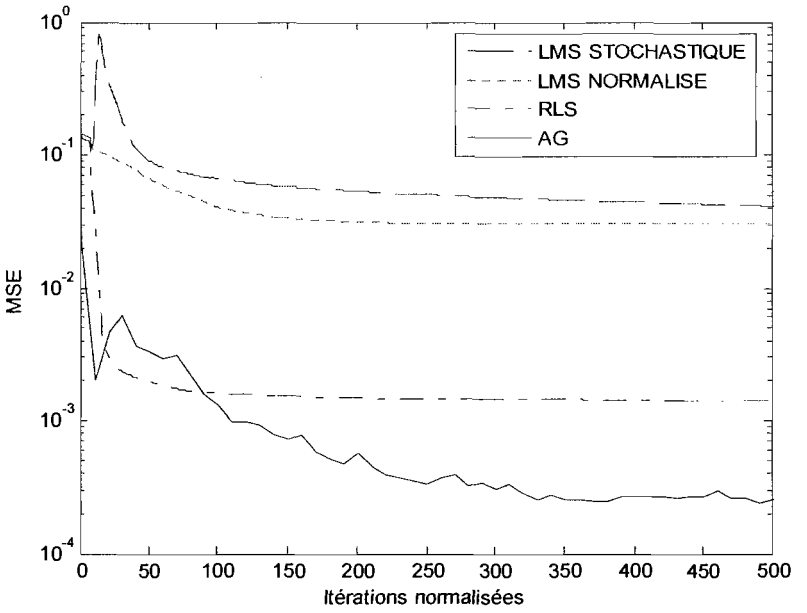

Figure 4.13 Identification du canal linéaire ARMA par LMS, NLMS, RLS et AG pour 24 bits

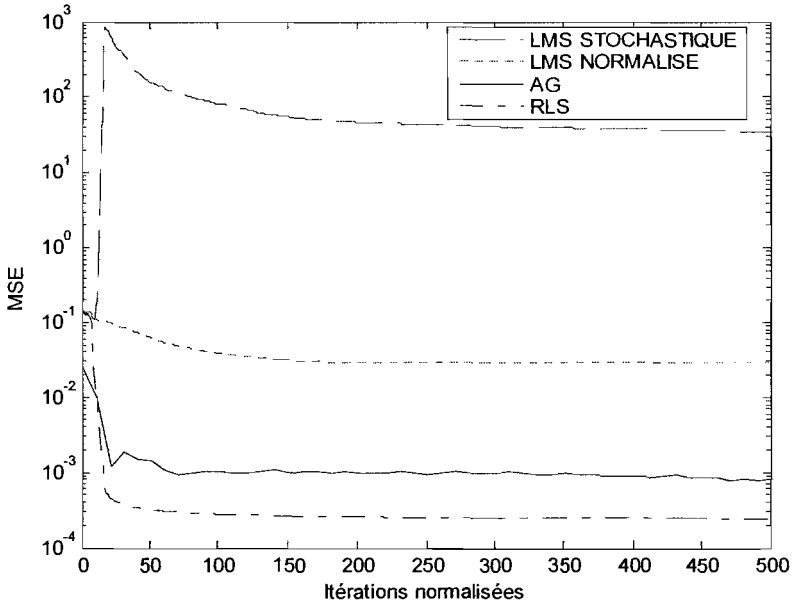

Figure 4.14 Identification du canal linéaire ARMA par LMS, NLMS, RLS et AG en virgule flottante

Au regard des figures 4.9 à 4.13, on remarque très rapidement que les AG sont plus performants que les autres méthodes, car quelque soit la longueur des nombres binaires utilisée le MSE est toujours inférieur à 10^{-3}. Seul le RLS est capable de rivaliser avec les AG pour l'identification du canal ARMA, cependant il a besoin d'avoir une longueur de bits minimale de 24 bits afin d'être compétitif (figure 4.14). De manière générale, les AG ont besoin de 350 itérations normalisées ou 35 générations de population afin d'atteindre une erreur asymptotique stable et de 80 itérations (8 générations) pour atteindre une erreur sous 10^{-3}.

Le tableau 4.2 ci-dessous, présente les paramètres du canal linéaire identifié par les différentes méthodes en comparaison des paramètres idéaux et pour diverses grandeurs de bits. Ce tableau démontre une fois de plus la robustesse des AG face à l'effet de quantification, car la méthode n'est pas affectée par le choix de la longueur du nombre binaire des paramètres. Ce tableau permet aussi de constater que les AG donnent de meilleurs résultats que les autres méthodes, particulièrement quand le nombre de bits diminue. Le LMS, le NLMS et le RLS éprouvent particulièrement beaucoup de difficulté dans l'identification des paramètres b_1 et b_2. Cette difficulté à identifier certains paramètres est liée à la complexité du canal.

Tableau 4.2 Identification des paramètres du canal linéaire

Méthodes	a1	a2	b1	b2	c0	c1	c2
Paramètres à identifier	1.5	-0.7	1	0.5	1	-1	0.2
	Virgule flottante						
LMS	1.49	-0.69	0.49	0.45	1	-0.99	0.19
NLMS	1.49	-0.69	0.40	0.32	1	-0.99	0.19
RLS	1.49	-0.69	0.99	0.51	1	-0.99	0.19
GA	1.37	-0.68	0.87	0.5	1	-1	0.17
	24-bits						
LMS	1.5	-0.7	0.99	1.09	1	-1	0.2
NLMS	1.49	-0.67	1.01	1.03	1	-0.99	0.2
RLS	1.49	-0.69	0.97	0.53	1	-0.99	0.19
GA	1.5	-0.66	1	0.5	1	-1.02	0.19
	16-bits						
LMS	1.33	-0.46	0.40	0.35	1.01	-0.69	0.14
NLMS	1.49	-0.66	0.89	0.75	1.01	-0.91	0.18
RLS	1.21	-0.45	0.97	0.87	1.01	-0.62	0.13
GA	1.5	-0.71	1	0.5	1	-1.01	0.17
	12-bits						
LMS	1.01	-0.36	0.40	0.42	1.03	-0.48	0.16
NLMS	1.51	-0.69	0.88	0.88	1.02	-1.04	0.16
RLS	1.06	-0.40	0.80	0.74	0.98	-0.49	0.13
GA	1.5	-0.70	1	0.5	1	-1.01	0.15
	8-bits						
LMS	1.40	-0.53	0.48	0.48	1	-0.5	0.15
NLMS	1.5	-0.73	0.82	0.76	0.98	-0.98	0.23
RLS	1.48	-0.68	0.82	0.53	0.96	-0.90	0.17
GA	1.5	-0.72	0.63	0.5	1	-0.99	0.12
	6-bits						
LMS	1.43	-0.06	0.18	0.18	0.93	-0.43	0.18
NLMS	1.5	-0.56	0.25	0.25	1	-0.81	0.31
RLS	1.5	-0.56	1.93	0.62	1	-1	0.12
GA	1	-0.69	1	0.5	0.73	-1	0.12

Afin de démontrer la reproductibilité et la fiabilité des AG, sa répétitivité ainsi que celle des autres méthodes ont été mises à l'épreuve. Les figures 4.15 à 4.18 présentent les variances de la moyenne des erreurs quadratiques moyennes (MSE) pour chacune des méthodes et pour diverses grandeurs de bits.

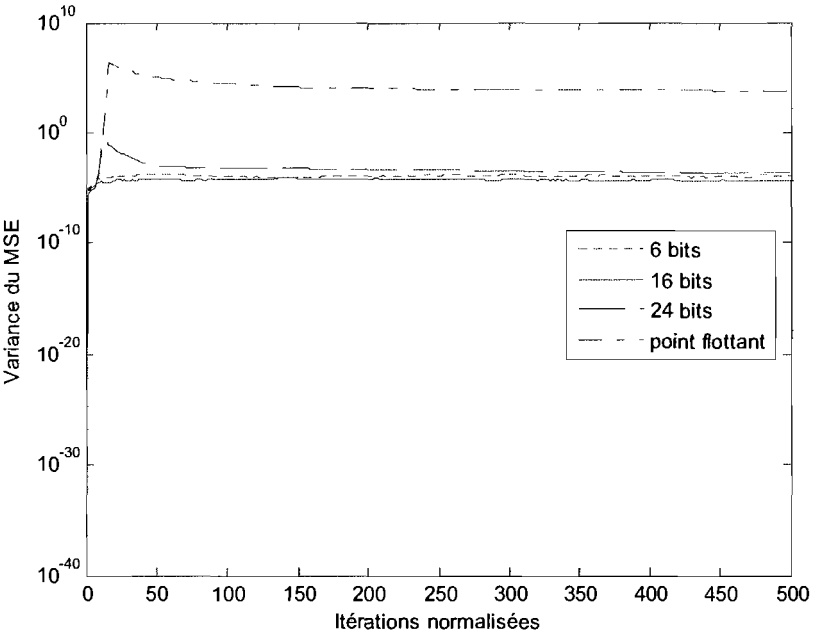

Figure 4.15 Variance du MSE pour le LMS et pour différentes grandeurs de bits

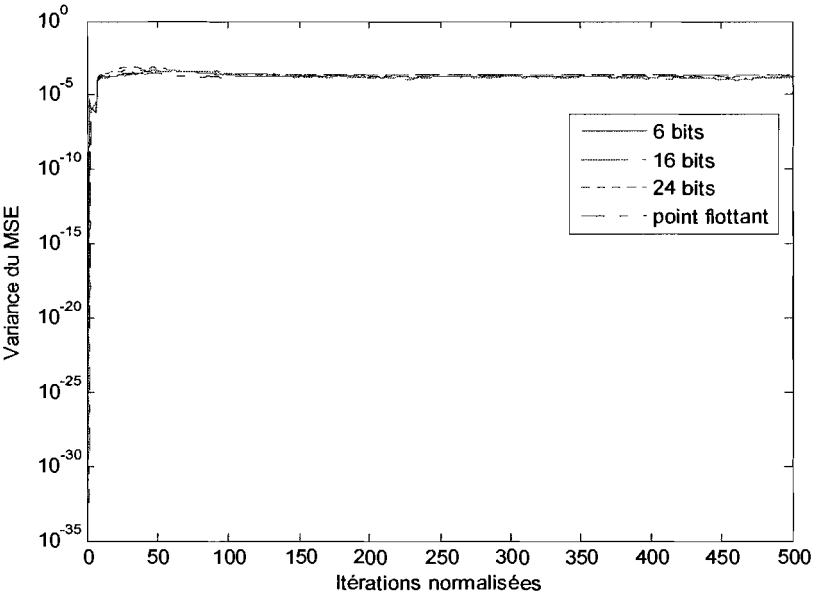

Figure 4.16 Variance du MSE pour le NLMS et pour différentes grandeurs de bits

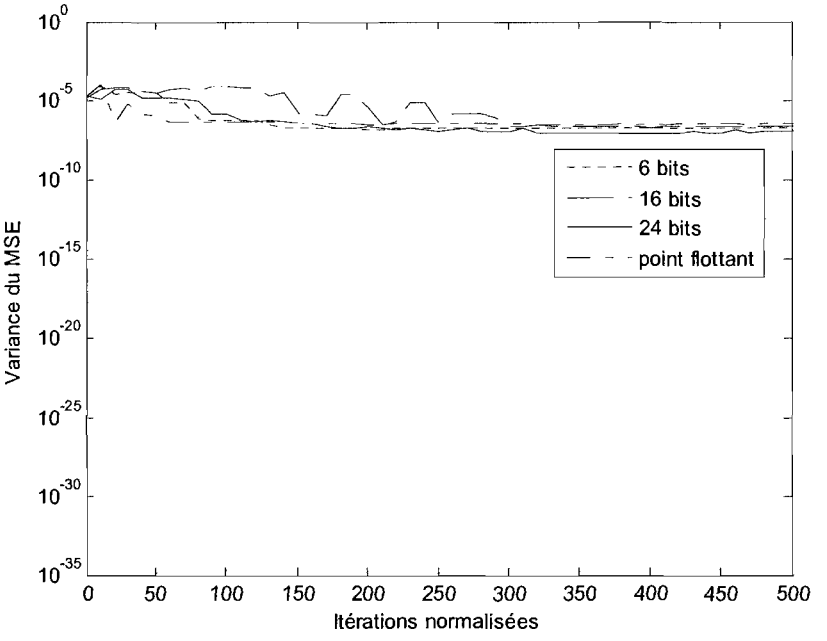

Figure 4.17 Variance du MSE pour l'AG et pour différentes grandeurs de bits

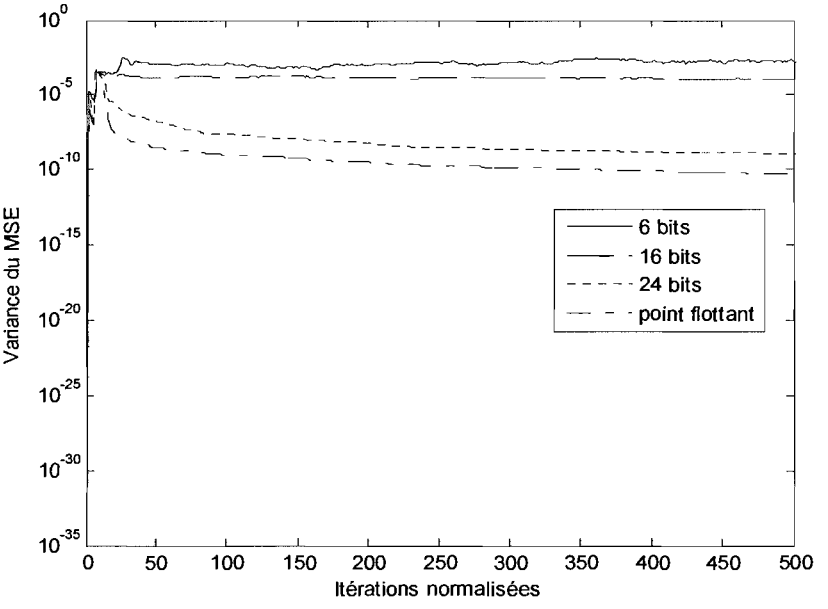

Figure 4.18 Variance du MSE pour le RLS et pour différentes grandeurs de bits

Les courbes de variance du MSE des figures 4.15 à 4.18 permettent de constater une excellente répétitivité des méthodes, mais aussi permet de voir que quelque soit le nombre

de bits utilisés, les AG démontrent une robustesse face à l'effet de quantification, car fournit une variance quasi identique peu importe les longueurs binaires employées. Cette étude de répétitivité permet d'observer comme plus haut que les méthodes classiques ont beaucoup de difficulté à évoluer avec très peu de bits.

4.4.2 Résultats de simulation pour le canal non linéaire

Les figures 4.19 à 4.24 présentent les résultats de simulation du canal non linéaire, identifié à l'aide du LMS, du NLMS, du RLS et des AG. Les figures sont réalisées pour diverses valeurs de bits, mais également en virgule flottante. Ces figures ont pour objectif de montrer l'incapacité des méthodes usuelles à identifier un canal non linéaire, mais aussi de montrer la robustesse des AG et surtout leur capacité à opérer avec très peu de bits.

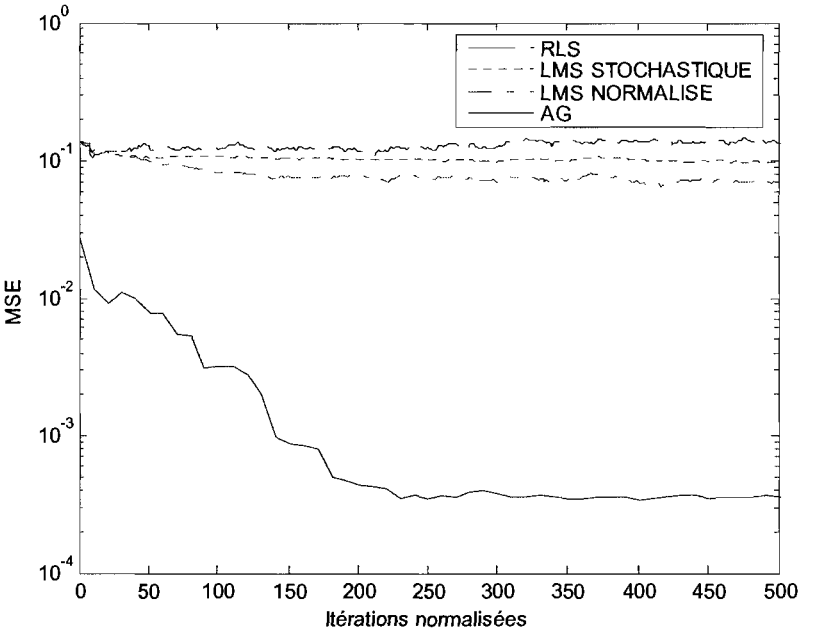

Figure 4.19 Identification du canal non linéaire ARMA par LMS, NLMS, RLS et AG pour 6 bits

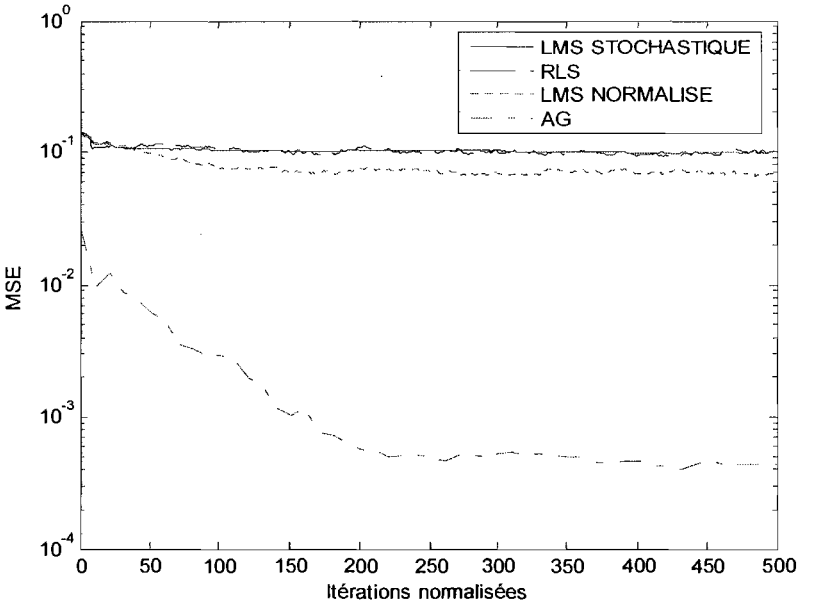

Figure 4.20 Identification du canal non linéaire ARMA par LMS, NLMS, RLS et AG pour 8 bits

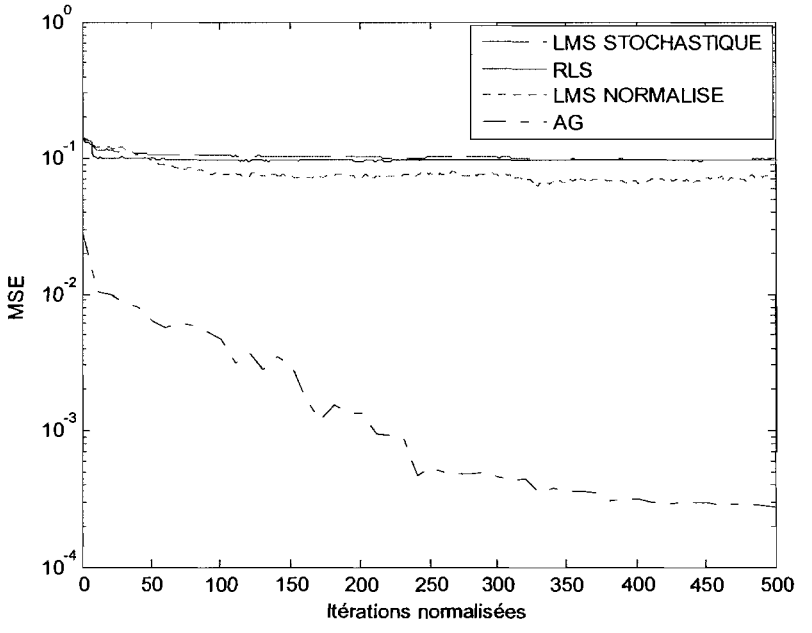

Figure 4.21 Identification du canal non linéaire ARMA par LMS, NLMS, RLS et AG pour 12 bits

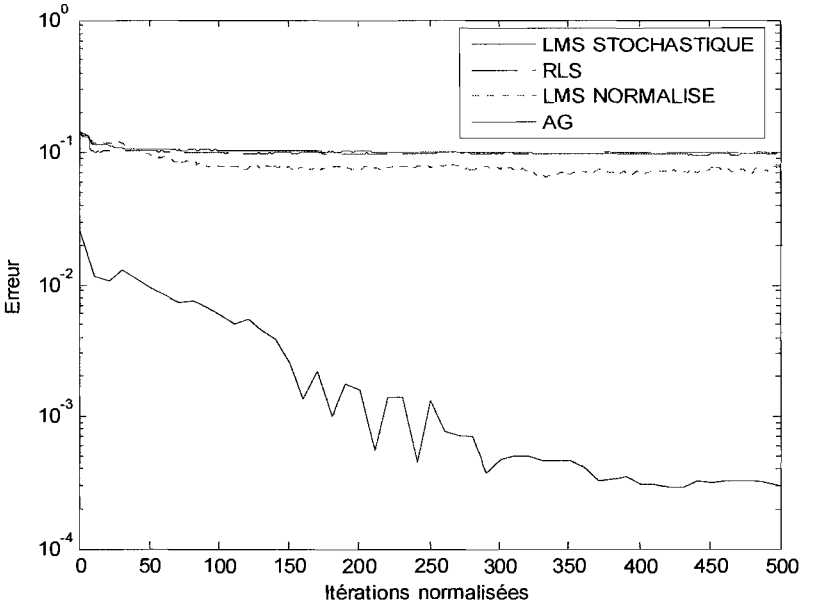

Figure 4.22 Identification du canal non linéaire ARMA par LMS, NLMS, RLS et AG pour 16 bits

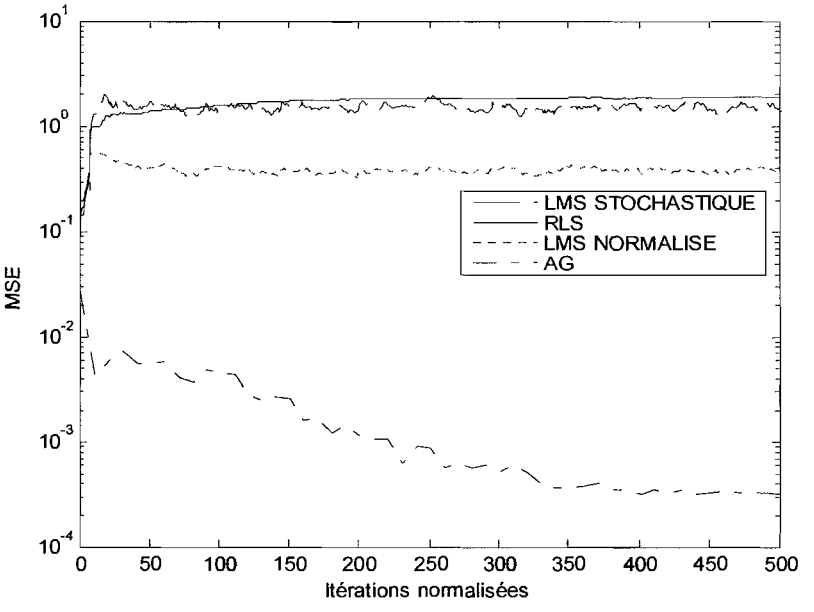

Figure 4.23 Identification du canal non linéaire ARMA par LMS, NLMS, RLS et AG pour 24 bits

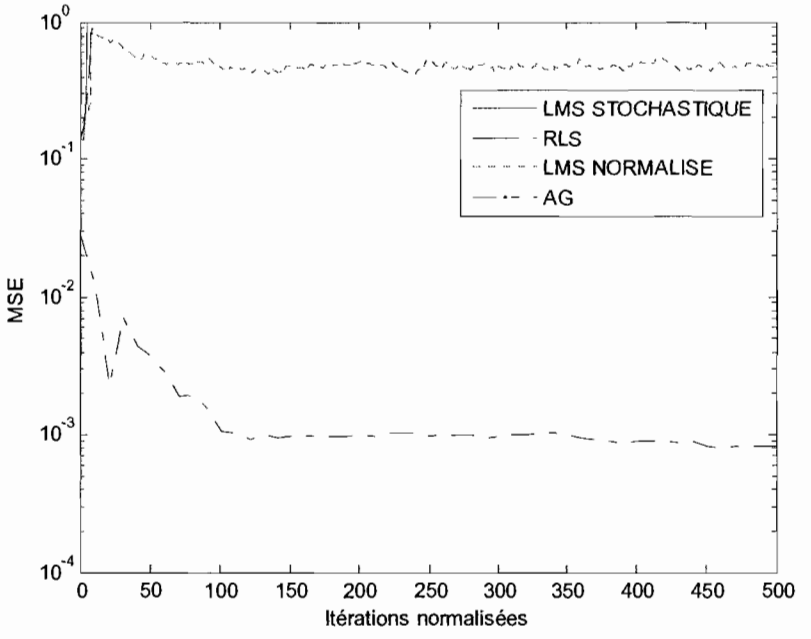

Figure 4.24 Identification du canal non linéaire ARMA par LMS, NLMS, RLS et AG en virgule flottante

Une analyse des figures 4.19 à 4.24 permet de constater que le LMS, le NLMS et le RLS sont inopérants pour le canal non linéaire. Seuls les AG sont capables d'identifier le canal non linéaire. Quelle que soit la grandeur de bit utilisée, les AG demeurent performants, avec un MSE toujours inférieur à 10^{-3}. Comme dans le cas linéaire, les AG ont besoin de 350 itérations normalisées ou 35 générations de population afin d'atteindre une erreur asymptotique stable et de 80 itérations (8 générations) pour atteindre une erreur sous 10^{-3}.

Le tableau 4.3, présente les paramètres du canal non linéaire identifié par les différentes méthodes en comparaison des paramètres idéaux et pour diverses grandeurs de bits.

Tableau 4.3 Identification des paramètres du canal non linéaire

Méthodes	a1	a2	b1	b2	c0	c1	c2
Paramètres à identifier	1.5	-0.7	1	0.5	1	-1	0.2
	Virgule flottante						
LMS	0.06	0.07	0	0	0.22	0.04	0.02
NLMS	1.04	-0.54	-0.07	0	8.84	-0.10	0.93
RLS	0.70	-0.18	5.28	138	5.91	-1.04	0.32
GA	1.5	-0.71	0.94	0.5	1	-0.98	0.17
	24-bits						
LMS	0.36	1.24	0	0	0.79	-0.003	0.44
NLMS	1.32	-0.79	0.218	0	3.07	0.74	-0.49
RLS	1.39	-0.93	6.96	19.63	3.68	-2.55	0.18
GA	1.5	-0.69	1	0.46	0.54	-1	0.19
	16-bits						
LMS	0.87	-0.10	0.52	0.51	1.48	-0.15	0.16
NLMS	1.34	-0.77	1.80	1.89	1.22	-0.57	0.17
RLS	0.71	-0.14	1.29	1.96	1.47	-0.07	0.12
GA	1.5	-0.7	1	0.5	1	-1	0.19
	12-bits						
LMS	0.96	-0.15	0.12	0.14	1.5	-0.21	0.13
NLMS	1.26	-0.68	0.28	0.30	1.18	0.28	0.40
RLS	0.65	-0.11	0.07	1.27	1.40	-0.06	0.13
GA	1.5	-0.68	1	0.5	1	-1.01	0.19
	8-bits						
LMS	0.92	-0.28	0.31	0.15	1.6	-0.21	0.18
NLMS	1.56	-0.53	0.82	0.10	1.32	-1	0.17
RLS	1.01	-0.28	1.82	0.43	1.76	-0.46	0.28
GA	1.5	-0.71	1	0.5	1	-1.01	0.16
	6-bits						
LMS	1.12	-0.31	0.87	0.75	1.56	-0.25	0.18
NLMS	1.56	-0.56	0.62	0.56	1.87	-0.43	0.18
RLS	1.12	-0.12	1.87	1.18	1.75	-0.31	0.18
GA	1.5	-0.69	1	0.5	1	-0.99	0.14

Le tableau 4.3 permet de constater l'inefficacité du LMS, le NLMS et le RLS dans l'identification du canal ARMA non linéaire. Ils ne sont capables d'identifier correctement aucun des paramètres du canal. On voit ici que malgré la non linéarité du canal, les AG fournissent des résultats qui sont quasi similaires à ceux obtenus avec le canal linéaire. La méthode des AG n'est pas affectée par la variation du nombre de bits des paramètres.

Comme dans le cas linéaire, la répétitivité de la méthode des AG a été étudiée. Les résultats obtenus sont identiques à ceux de la figure 5.17. Idem au cas linéaire, les courbes de variance du MSE permettent de constater une excellente répétitivité de la méthode des

AG, elles permettent aussi de voir que quelque soit le nombre de bits utilisés, les AG démontrent une grande robustesse face à l'effet de quantification, car fournissent une variance quasi identique peu importe les grandeurs bits employées.

4.5 Étude de la complexité

4.5.1 Détail du calcul de complexité des AG

Cette section présente le détail du calcul de complexité des AG à travers l'estimation des opérations arithmétiques (addition, multiplication, division) effectuées pour une génération. Les éléments à surveiller lors de l'évaluation de la complexité de la méthode sont : la taille de la population (P), le nombre de coefficients du filtre (M), le pourcentage de la population affecté par la recombinaison (p_c), le pourcentage de la population affectée par la mutation (p_m) et la taille de la fenêtre du calcul de l'erreur (d).

4.5.2 Comparaison de la complexité

Le tableau 4.4 ainsi que les figures 4.25 à 4.27 fournissent les résultats des études de complexité des diverses méthodes. L'étude de complexité du LMS (NLMS) et du RLS est connue dans la littérature depuis plusieurs décennies [20]. La comparaison de complexité entre le LMS, le RLS et les AG en plus d'être faite pour les opérations arithmétiques, est également faite en terme de "Full Adder". Le "Full Adder" est l'élément de base des circuits arithmétiques tels que les additions, les multiplications, les divisions, les exponentiels, etc.

Tableau 4.4 Étape de calcul de la complexité des AG

Étape	Équation	Addition/ Soust.	Mult.	Div.	Autres
Évaluation de la population	3.4	(M-1)P	MP		
Erreur	3.10	*d*P			
Norme de l'erreur		*(d-1)P*	*dP*	*P*	*Pour d>1, sinon 0*
fonction sélective	3.9			P	
Somme des fonctions sélectives	3.13	P-1			
Probabilité de sélection	3.14			P	
Probabilité cumulative	3.15	P-1			
Recombinaison					
Réévaluation de la population		(M-1)PP_c	MPP_c		
Mutation					
Réévaluation de la population	3.4	(M-1)PP_m	MPP_m		
TOTAL (d= P_c=P_m =1):		(3M-1)P-2	3MP	2P	

Le "Full Adder" est un circuit logique qui prend trois entrées binaires *A, B, CI* et fournit deux sorties binaires *S, CO*. Le nombre binaire (*CO, S*) est la représentation binaire de la somme arithmétique des entrées. Les équations logiques de bases des sorties *S et CO* peuvent être décrites selon les fonctions logiques suivantes:

$$S = A \oplus B \oplus CI \tag{5.1}$$

$$CO = A \cdot B + A \cdot CI + B \cdot CI \tag{5.2}$$

Le Tableau 4.5 ci-dessous présente une étude de complexité des différentes méthodes en terme d'opérations arithmétiques. En se basant sur le nombre de paramètres à identifier pour notre système ARMA (M=7) et sur ceux de la méthode des AG (population P=10, p_m=1 et p_c=1), on arrive à la conclusion selon laquelle pour notre application, les AG possèdent une complexité plus élevée en terme d'opérations arithmétiques par itération.

Tableau 4.5 Comparaison de la complexité des méthodes en terme d'opérations arithmétiques

Methodes	Opérations arithmétiques par itération			M=7, P=10, β=α=1
	Add/Soust.	Mult	Div	Nb op./it.
LMS	2M	2M+1	0	29
RLS	$2M^2+6M+4$	$2M^2+7M+5$	M^2+4M+3	292
GA	3MP-P-2	3MP	2P	428

Pour la même étude de complexité réalisée en terme de "Full Adder" dans le tableau 4.6 ci-dessous, on constate que le RLS est de très loin plus complexe que les AG et le LMS. Le calcul du nombre d'opérations en terme de "Full Adder" tient compte du nombre de bits de chacune des méthodes qui donnent les meilleurs résultats d'identification (Fig 4.9 à Fig 4.13). Des courbes de la section 5.2.2 on tire le nombre de bits rentrant dans le calcul du nombre d'opérations en terme de "Full Adder" pour chacune des méthodes. On a ainsi respectivement Q_LMS = 16, Q_RLS = 24 et Q_AG = 8 bits pour le LMS, le RLS et les AG.

Table 4-6 Comparaison de la complexité des méthodes en terme de Full Adder

Methodes	Full Adder par itération			M=7, P=10, β=α=1
	Add/Soust.	Mult	Div	Full Adder/iteration
LMS	2M	$(2M+1)Q\_LMS^2$	0	224+3840=4064
RLS	$2M^2+6M+4$	$(2M^2+7M+5)Q\_RLS^2$	$(M^2+4M+3)Q\_RLS^3$	5808+87552+1105920=1199280
GA	3MP-P-2	$(3MP)Q\_AG^2$	$(2P)Q\_AG^3$	1584+13440+10240=25264

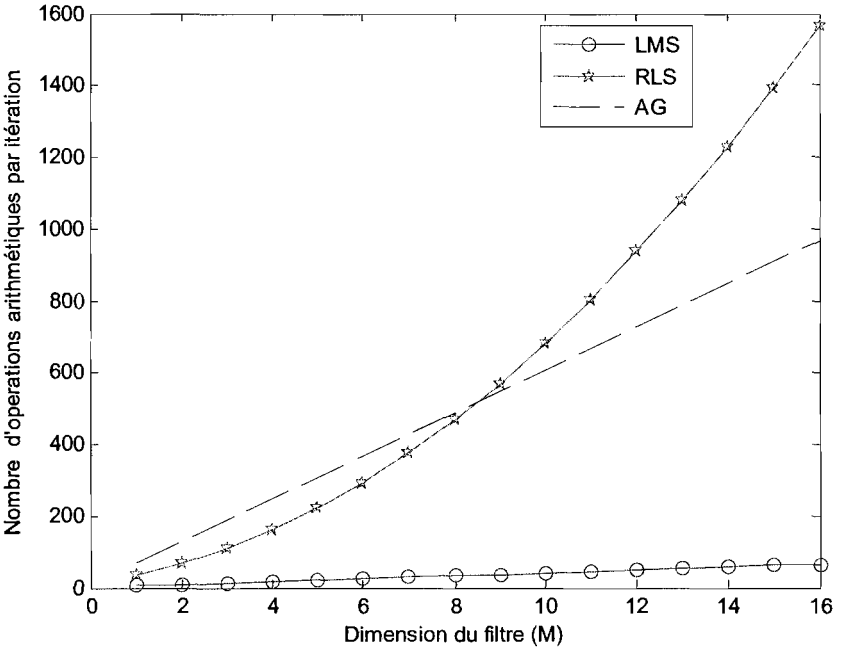

Figure 4.25 Nombre d'opérations arithmétiques en fonction de la dimension du filtre

La figure 4.25 présente le nombre d'opération arithmétique (additions, soustraction, multiplication et division) pour chaque génération. Il permet de constater que dans les conditions de fonctionnement de notre système, les AG sont plus complexes que le RLS et le LMS. On constate également que plus la taille du filtre (nombre de paramètres à identifier) augmente, plus la complexité des méthodes croît. Lorsque le nombre de

coefficients du filtre dépasse 9, la complexité du RLS augmente de façon exponentielle et devient plus grande que celle des AG, en terme d'opérations arithmétiques. Des trois méthodes évaluées, le LMS est de loin le moins complexe en nombre d'opérations arithmétiques.

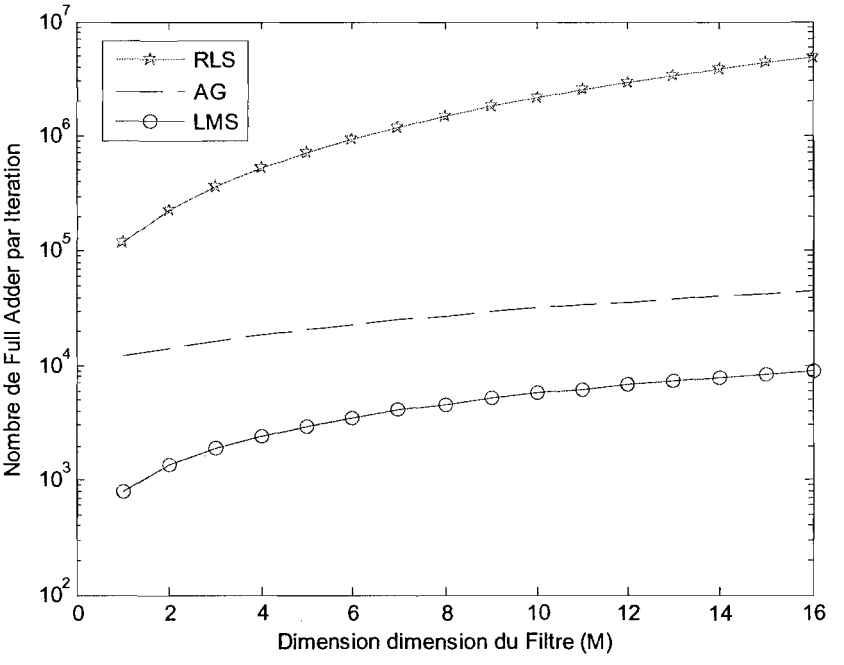

Figure 4.26 Nombre de Full Adder par itération en fonction de la dimension du filtre

En observant la figure 4.26, on remarque que la complexité des méthodes croît autant que la taille du filtre croit. Comme présenté au tableau 4.6, lorsque l'étude de complexité est réalisée en terme de "Full Adder" par itération ou génération, le RLS est de très loin plus complexe que les AG et le LMS. À noter que le nombre de bits de quantification des méthodes est respectivement de 16-bits, 24-bits et 8-bits pour le LMS, RLS et AG. Ces longueurs binaires correspondent aux meilleures performances de chacune de ces méthodes.

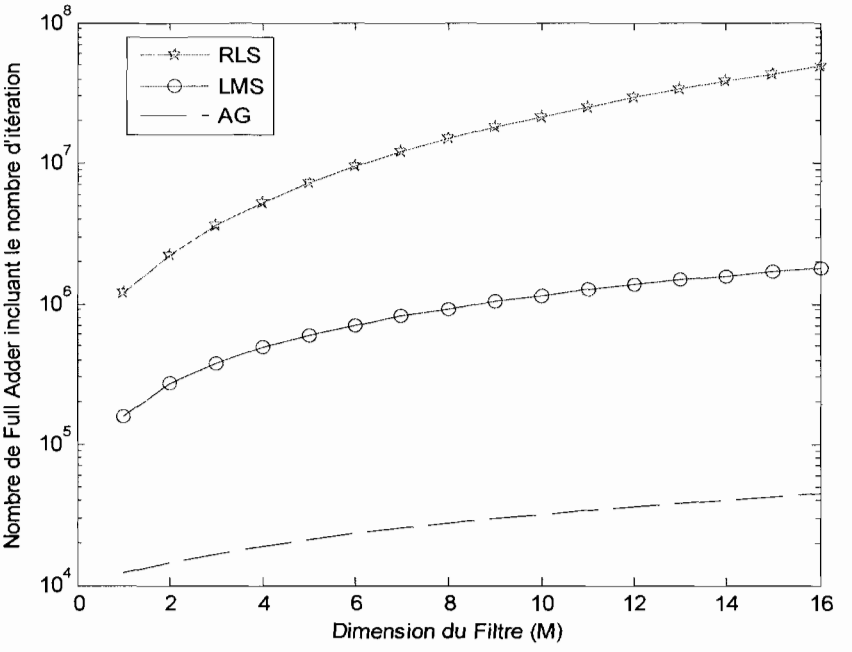

Figure 4.27 Nombre Full Adder incluant le nombre d'itération pour atteindre un MSE de 3^{e}-2 en fonction de la dimension du filtre

En conservant les mêmes conditions de simulations que celles de la figure 4.26, la figure 4.27 permet de déterminer le nombre de "Full Adder" incluant le nombre d'itérations pour atteindre une erreur de $3x10^{-2}$. Ce niveau d'erreur $3x10^{-2}$ est choisis comme cible car à partir de ce niveau d'erreur les coefficients estimés sont proche des coefficients réels. En observant les figures de la section 4.22 et en conservant le nombre de bits mentionné plus haut pour chacune des méthodes, nous avons pu déterminer le nombre d'itérations qui permet d'atteindre l'erreur cible de $3x10^{-2}$. On a ainsi respectivement pour le LMS, le RLS et les AG, 200, 10 et 1 itérations. L'analyse de la figure 4.27 permet d'affirmer qu'en terme de nombre de "Full Adder" nécessaire afin d'atteindre l'erreur cible de $3x10^{-2}$, les AG sont de loin les moins complexes suivi du LMS et du RLS. On note également comme dans les deux figures précédentes que toute augmentation de la dimension du filtre entraîne l'augmentation de complexité des méthodes.

4.6 Discussion

Le chapitre présent nous a permis d'observer et d'analyser les résultats de simulation. Les simulations observées plus haut dans ce chapitre, sont les résultats de la mise en œuvre des équations des méthodes vus dans les chapitres 1 et 3. Compte tenu de l'importance que revêt le choix des paramètres de réalisation d'une méthode, une étude détaillée des paramètres des AG a été effectuée. Cette étude a permis de faire le choix des paramètres les mieux adaptés pour notre application. Par la suite nous avons établi les critères de comparaison des différents algorithmes, afin de bien spécifier les conditions de simulations.

L'identification du canal linéaire a démontré que la méthode basée sur les AG était la plus performante, cela, peu importe, la grandeur de bit utilisé. Des autres méthodes, seul le RLS a fourni des résultats à même de rivaliser avec les AG. Cependant afin de prétendre obtenir des résultats proches ou supérieurs à ceux des AG, le RLS doit se prévaloir d'une longueur de bit minimale de 24 bits. De manière générale nous avons pu constater que les AG ont besoin de 350 itérations normalisées ou 35 générations de population afin d'atteindre une erreur asymptotique stable. Nous avons également pu remarquer que Le LMS, le NLMS et le RLS éprouvent beaucoup de difficulté dans l'identification des paramètres b_1 et b_2. Cette difficulté à identifier certains paramètres est liée à la complexité du canal. La répétitivité des AG, ainsi que celle des autres méthodes a été évaluée. L'étude de la variance a permis de constater une excellente répétitivité des méthodes, les AG démontrent grande robustesse face à l'effet de quantification, car fournit une variance quasi identique peu importe les grandeurs bits employées.

Au niveau du canal non linéaire, force est de constater l'inefficacité et l'incapacité des méthodes usuelles à identifier les paramètres du canal ARMA. Ici seuls les AG sont

opérationnels. Comme dans le cas linéaire, les AG sont d'une grande robustesse face à l'effet de quantification, ils sont capables d'identifier le système, même avec très peu de bits. Ils ont comme dans le cas linéaire besoin de 350 itérations normalisées ou 35 générations de population afin d'atteindre une erreur asymptotique stable. La répétitivité des AG a été évaluée, afin de montrer que la méthode n'est pas caduque. L'étude de la variance a permis de constater une excellente répétitivité de la méthode. Les AG démontrent une grande robustesse face à l'effet de quantification, car fournissent une variance quasi identique peu importe les grandeurs bits employées.

La complexité des méthodes a également été étudiée dans ce chapitre. Le détail du calcul de complexité de la méthode des AG pour une génération a été effectué. Cette étude nous a permis de comparer la complexité des AG avec celle du LMS et du RLS. En terme d'opérations arithmétiques par itération et selon les paramètres des AG choisis (P=10, p_m=1 et p_c=1) nous avons pu constater que les AG sont plus complexes. Cependant en terme de "Full Adder" par génération, le RLS devient plus complexe, suivi des AG et du LMS. Nous avons également comparé les méthodes en fonction du nombre de "Full Adder" nécessaire pour atteindre un MSE de $3x10^{-2}$. Dans ce dernier exercice les AG étaient la méthode la moins complexe suivie du LMS et du RLS. De manière générale, nous avons noté qu'une augmentation de la dimension du filtre provoque irrémédiablement une augmentation de la complexité des méthodes.

Chapitre 5 - Implémentation des algorithmes génétiques dans l'environnement Simulink

Les résultats obtenus dans le chapitre précédent, démontrent que la méthode basée sur les AG donne les meilleurs résultats pour l'identification du système ARMA linéaire et non linéaire. Compte tenu de la robustesse des AG, ce chapitre propose une implantation des AG dans l'environnement Simulink®. Cette implémentation dans l'environnement Simulink® pourrait être le prémisse pour une future implémentation sur une plateforme de Xilinx. Cette implémentation sur une plateforme Xilinx pourrait permettre d'estimer les ressources matérielles (nombre de additionneurs, de multiplieurs, de registres, de soustracteurs, de cellules logiques, de tampons, etc.) des AG sur les composantes programmables FPGA (*Field Programmable Gate Array*), dans le but de choisir le meilleur FPGA qui répond au compromis coût/performance.

La première section de ce chapitre présentera le détail de la mise en œuvre des AG dans l'environnement Simulink. Dans la seconde section de ce chapitre, les résultats obtenus sous Simulink seront présentés.

5.1 Implémentation dans l'environnement Simulink

Simulink est l'extension graphique de Matlab permettant de représenter les fonctions mathématiques et les systèmes sous forme de diagramme en blocs. En suivant la logique de

réalisation des algorithmes génétiques énoncée au chapitre 4, les algorithmes génétiques seront convertis en blocs Simulink.

5.1.1 Génération de la population initiale

La création de la population initiale est réalisée à l'intérieur du bloc dénommé génération. La population est créée en utilisant des blocs de générations de chiffres aléatoires. Chaque individu de la population est issu d'une concaténation de 7 blocs de générations de chiffres aléatoires qui à chaque itération génère de nouveaux chiffres.

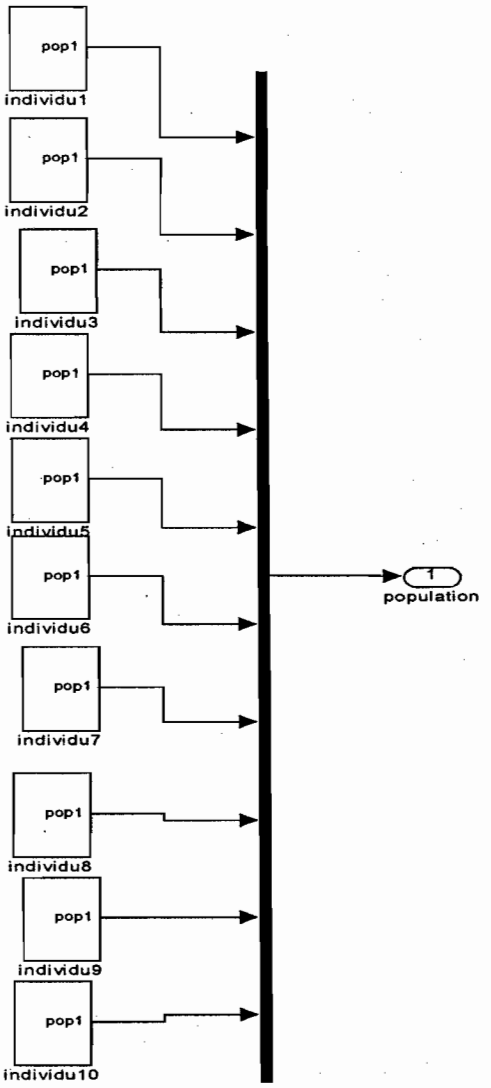

Figure 5.1 Génération de la population

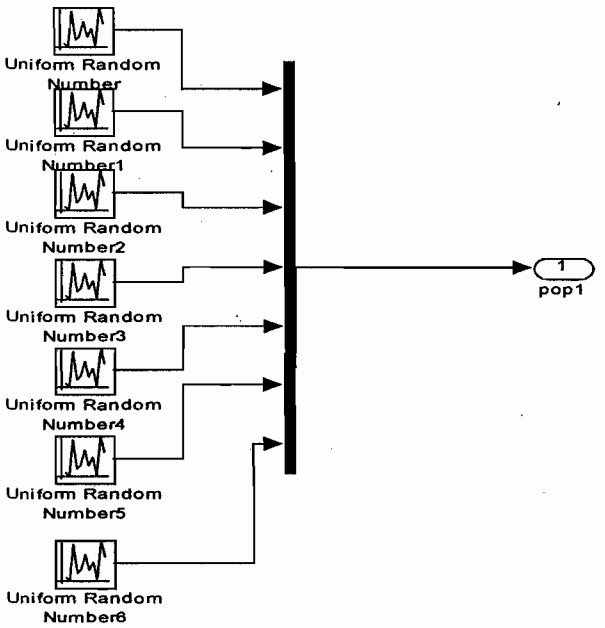

Figure 5.2 Création d'un individu

5.1.2 Calcul de l'erreur et évaluation de la population

Le calcul de l'erreur est effectué en s'inspirant de l'équation 3.10 tandis que l'évaluation de la population est réalisée grâce à l'équation 3.9. Cependant avant d'effectuer le calcul de l'erreur la population passe à travers le canal selon l'équation 3.4 dans le cas linéaire et 3.7 dans le cas non linéaire. La figure ci-dessous montre le détail de la réalisation des opérations de passage à travers le canal et du calcul de l'erreur.

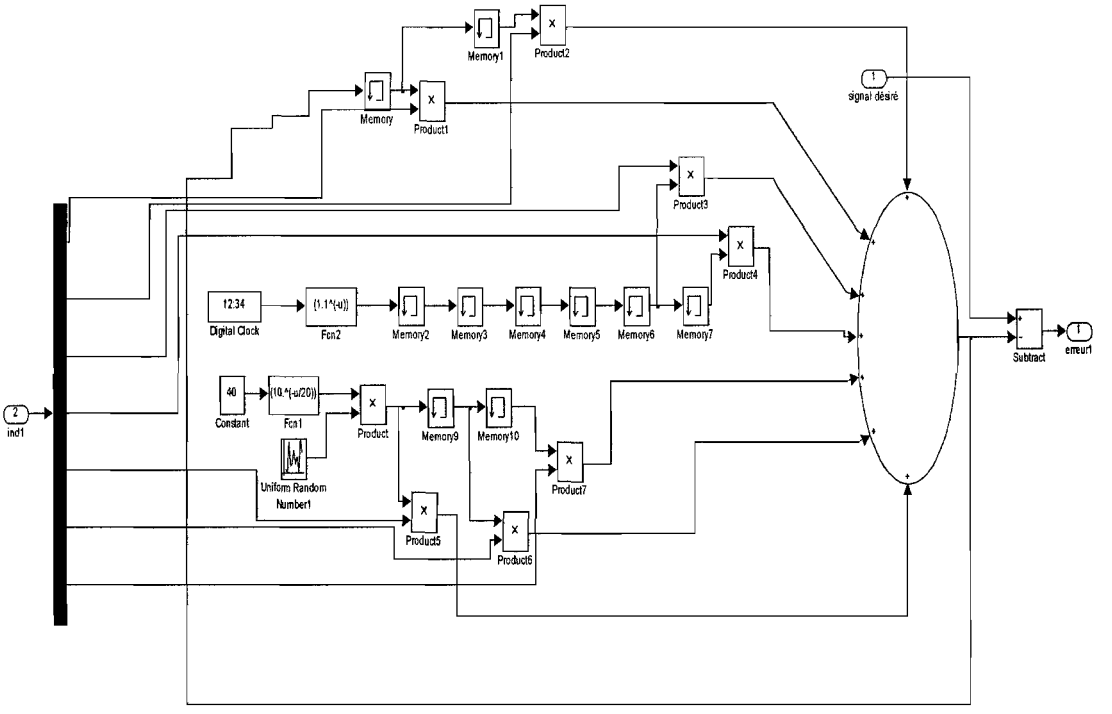

Figure 5.3 Passage à travers le canal et calcul de l'erreur

5.1.3 Sélection

La sélection est le processus qui permet de faire le choix des individus appelés à former la prochaine génération. Le mode de sélection adopté pour la résolution de notre problème d'identification est la sélection par la roulette décrite à la section 3.3.3. La probabilité de sélection de chacun des individus de la population est montrée à la figure 5.4.

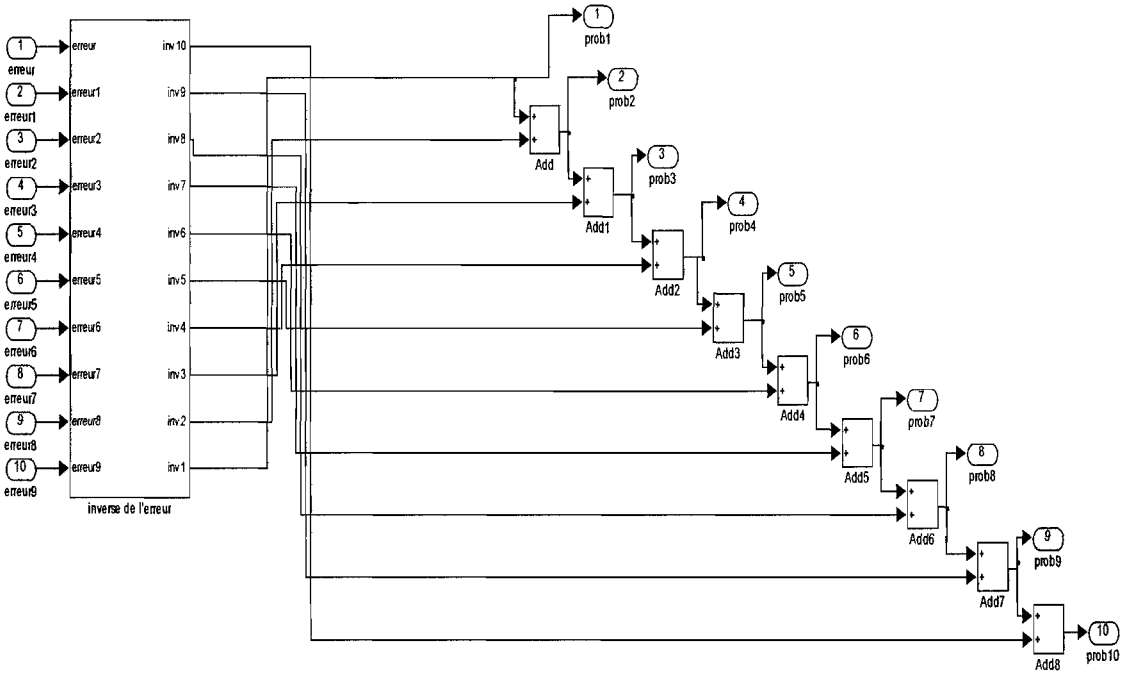

Figure 5.4 Calcul de probabilité de sélection de chaque individu

Après le calcul de probabilité, la roulette est tournée un nombre de fois égal à la taille de la population. À chaque tour de la roulette un individu est sélectionné. La figure 5.5 ci-dessous décrit le choix d'un individu.

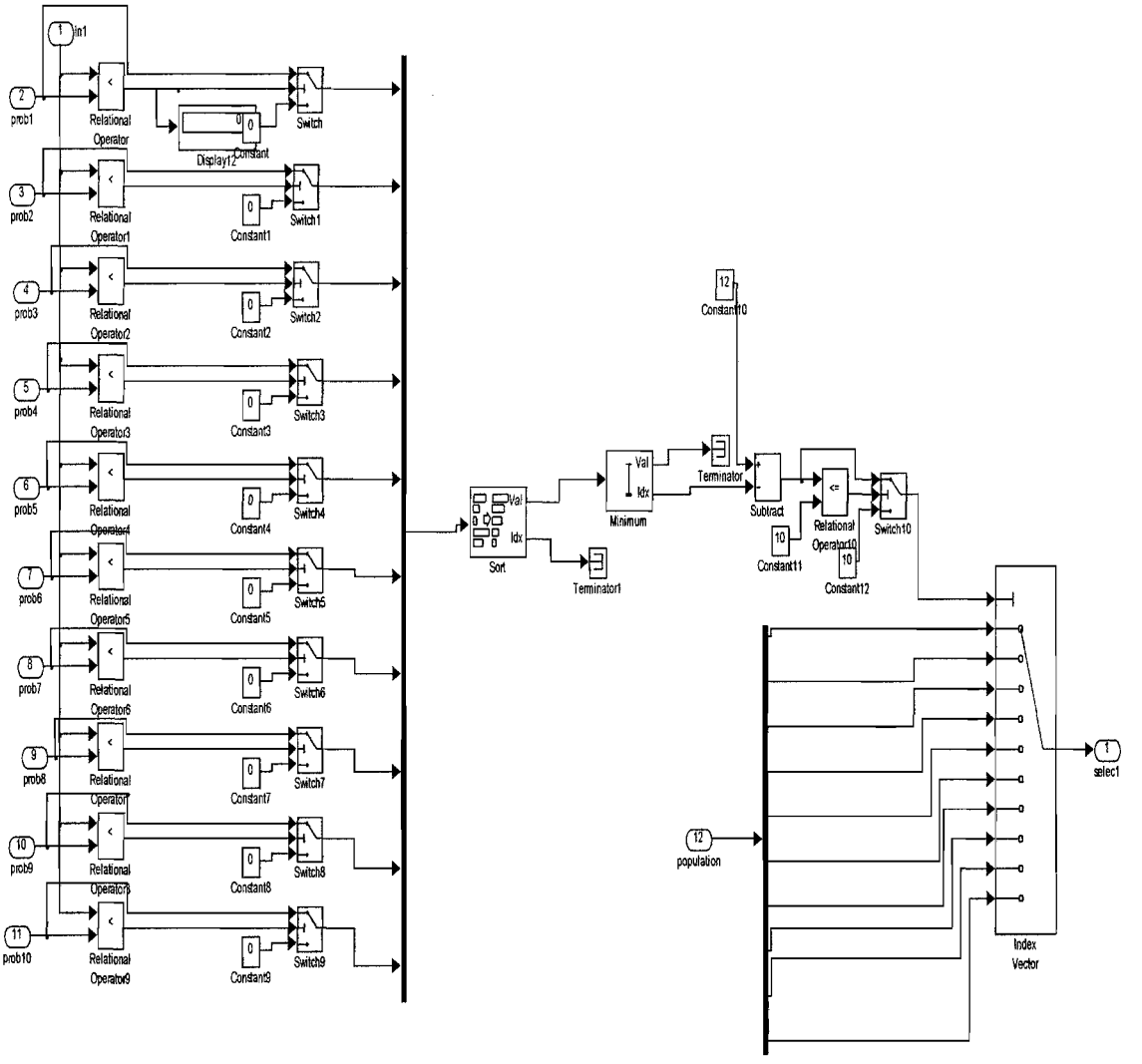

Figure 5.5 Sélection d'un individu

5.1.4 Recombinaison et mutation

Les opérateurs génétiques utilisés ici sont de types arithmétiques. Leurs modes de fonctionnement sont décrits à la section 2.6.1.1 et 2.6.1.2. Les figures 5.6 et 5.7 ci-dessous présentent le détail de leur réalisation.

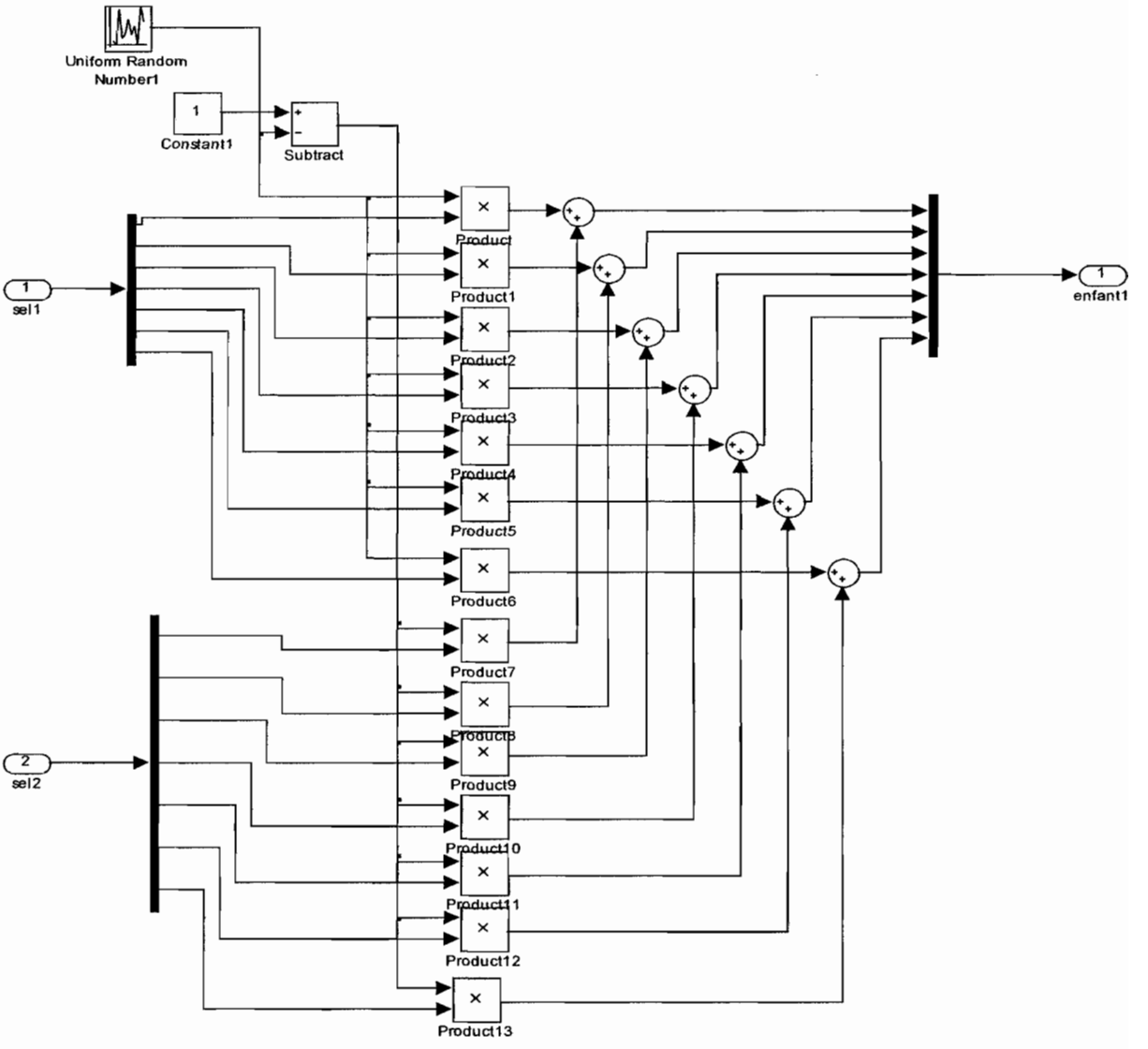

Figure 5.6 Recombinaison

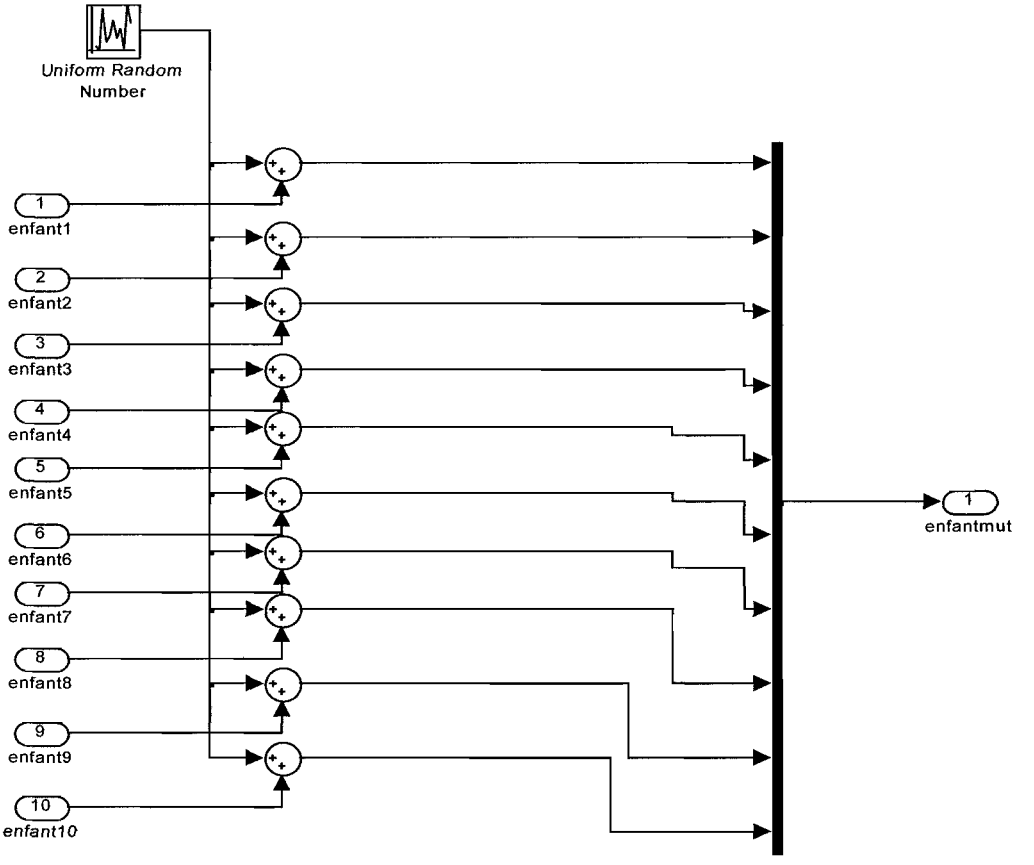

Figure 5.7 Mutation

5.1.5 Réinsertion et calcul du MSE

La réinsertion se fait au niveau du bloc réinsertion, elle permet l'utilisation de la population initiale en tout début et par la suite donne l'autorisation à la population issue de la mutation d'accéder au processus d'évaluation.

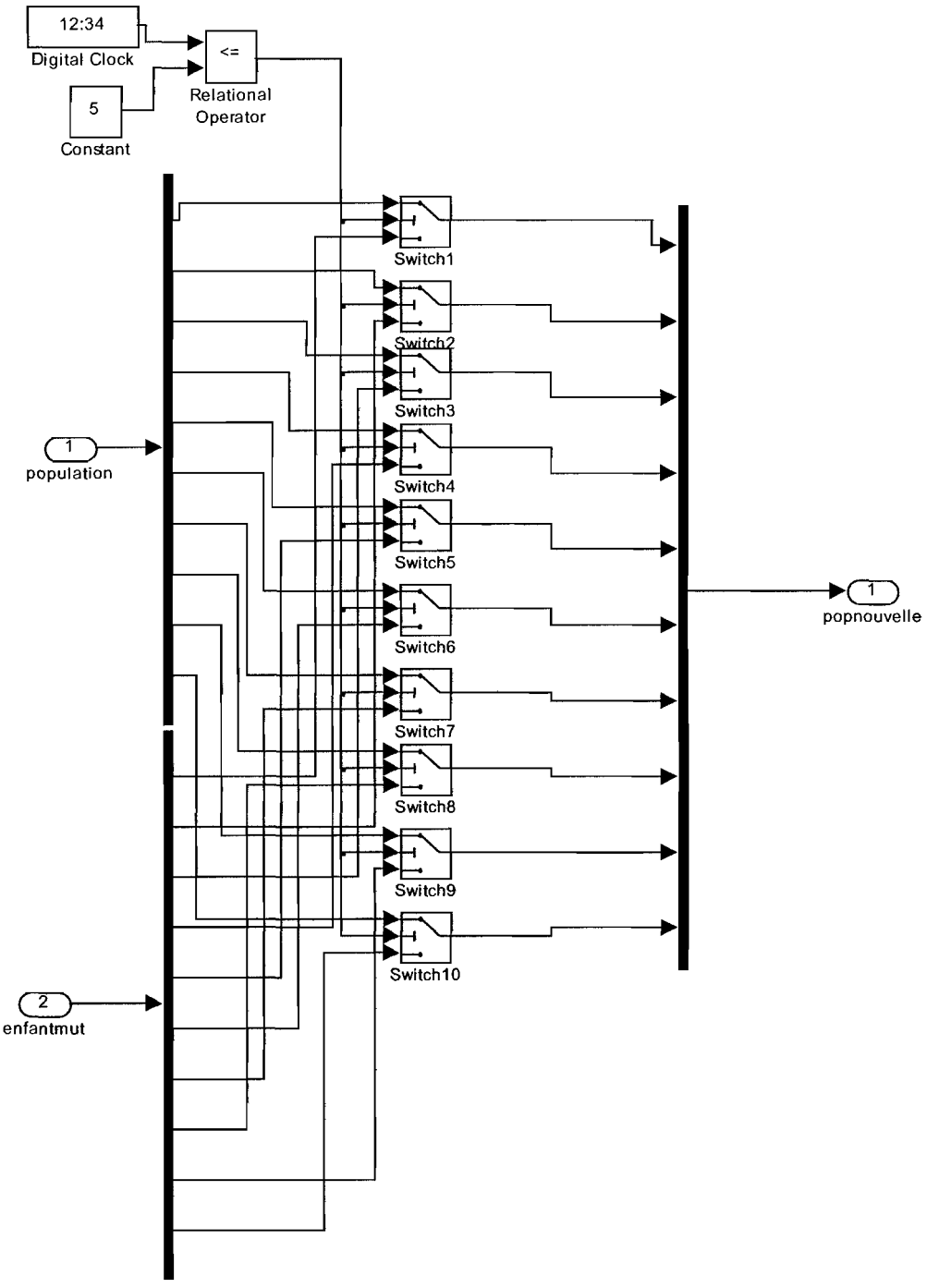

Figure 5.8 Réinsertion

L'intégration du calcul du MSE dans la plateforme, suit la même logique de calcul que l'équation 3.6.

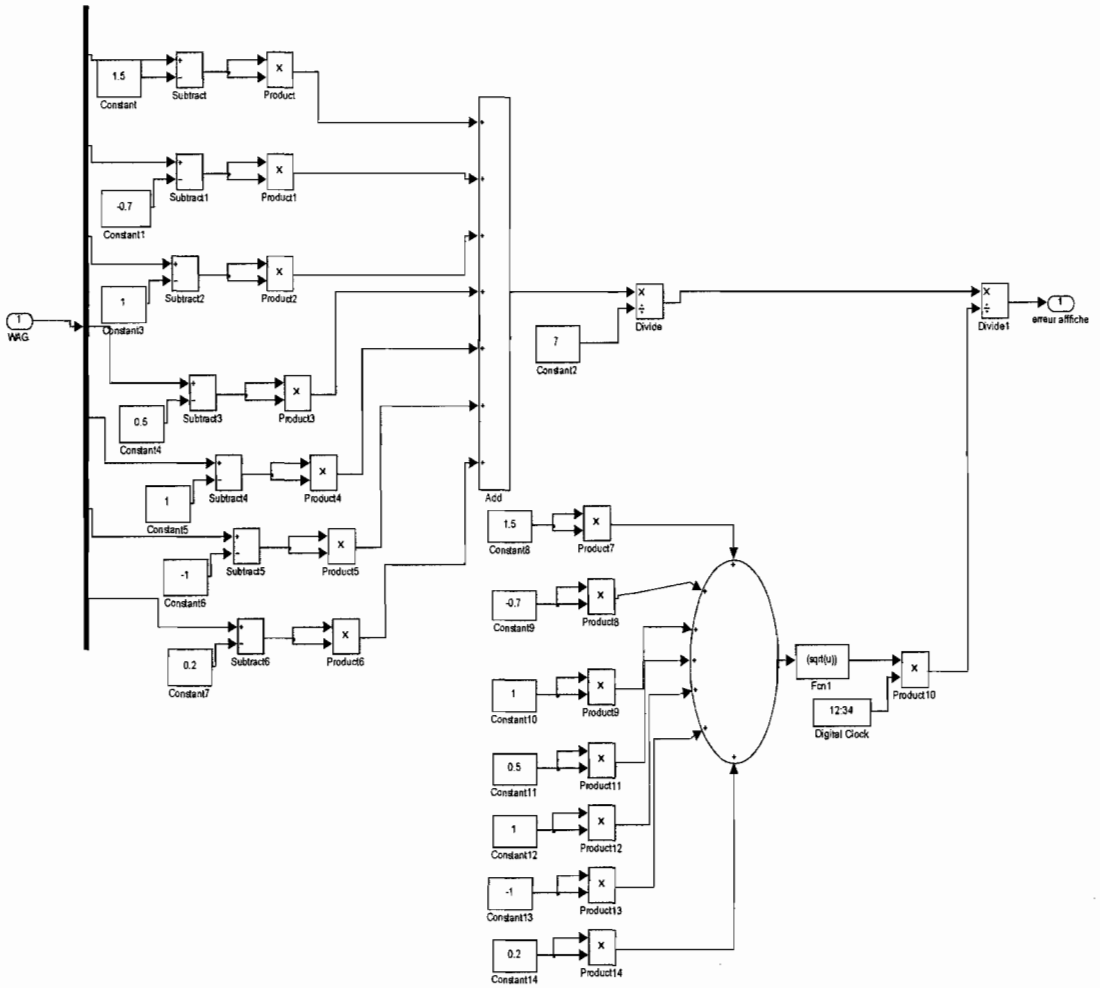

Figure 5.9 Calcul du MSE

Tous les blocs présentés plus haut, mis ensemble permettent de créer une plateforme sous Simulink basée sur les AG. La figure ci-dessous montre tous les blocs assemblés

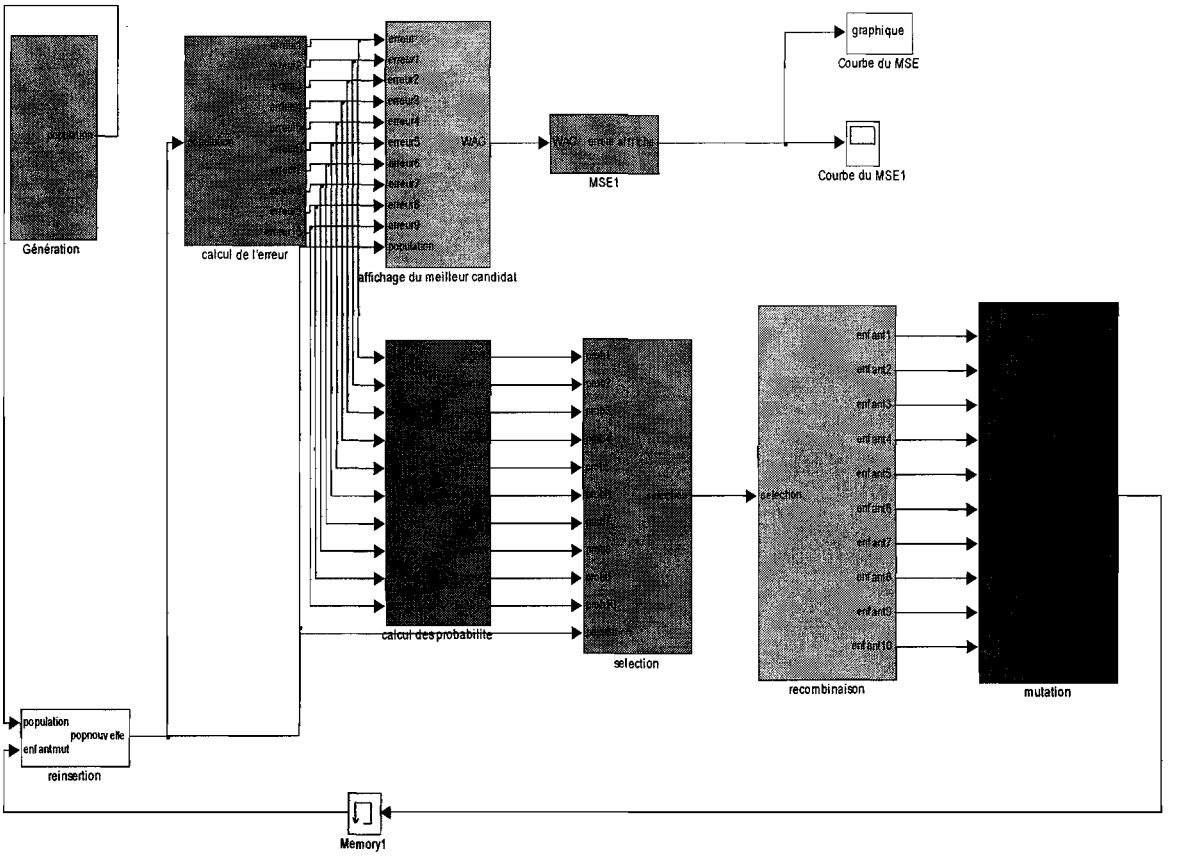

Figure 5.10 Représentation des AG sous Simulink

5.2 Résultats de l'AG sous Simulink

La section précédente nous a permis de voir dans le détail, le transfert d'un AG sous Simulink. Nous avons obtenu des résultats de simulation de l'AG proposé sur Simulink pour l'identification de paramètres des modèles ARMA de la section 3.2. Les courbes sont obtenues pour 1 répétition, sur 500 générations avec un niveau de bruit élevé, SNR = 0dB.

5.2.1 Modèle ARMA linéaire

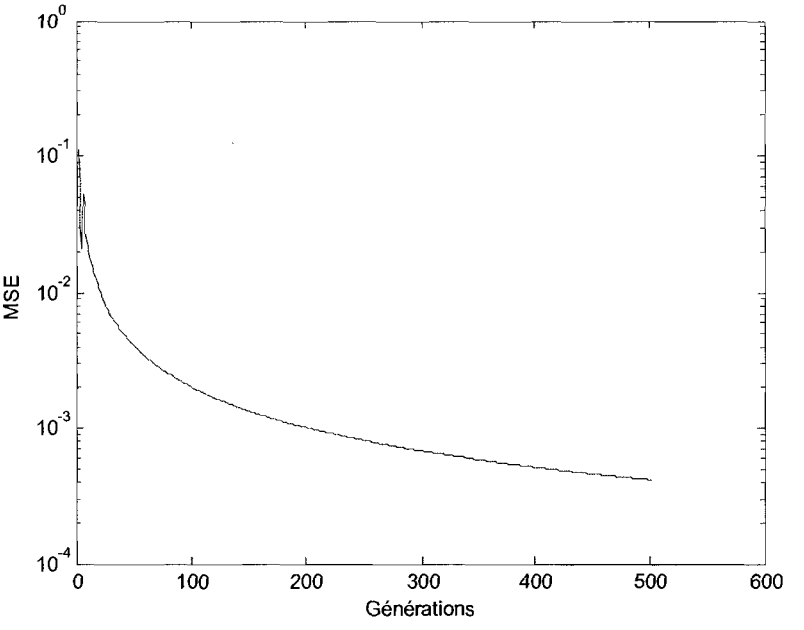

Figure 5.11 Identification du modèle ARMA linéaire sous Simulink

Comme dans le cas des simulations effectuées sous Matlab, les AG fournissent d'excellents résultats.

5.2.2 Modèle ARMA non linéaire

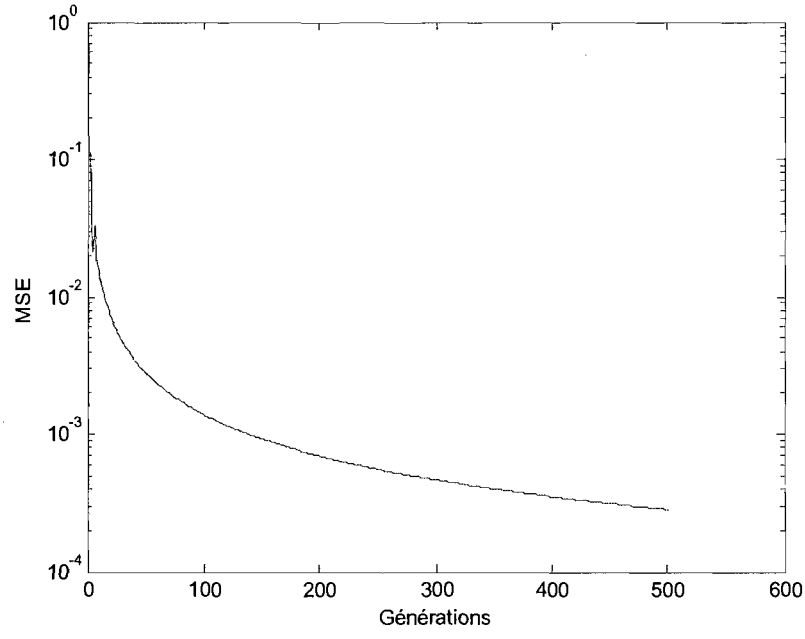

Figure 5.12 Identification du modèle ARMA canal non linéaire sous Simulink

Idem au cas linéaire l'identification d'un canal non linéaire sous Simulink donne un très bon résultat.

Nous constatons de manière générale que ce soit pour le canal linéaire ou non linéaire, les résultats de simulations obtenus dans Simulink sont très proches de ceux obtenus avec Matlab.

Chapitre 6 - Conclusion

Dans les systèmes de communications sans fil numériques modernes, l'identification de canaux (systèmes) est une application essentielle, très souvent utilisée. La connaissance des paramètres du canal de communication permet de faire une très bonne estimation des données transmises et ainsi d'avoir un taux d'erreur presque nul. De nombreuses méthodes et algorithmes dédiés à l'identification des systèmes ont été développés depuis des décennies. Compte tenu de la mobilité dans le temps des canaux de communication, il est utile d'avoir un algorithme qui peut identifier les paramètres du canal de manière adaptative. Afin de répondre à ce besoin d'adaptation du filtre, les algorithmes adaptatifs tels que : l'algorithme du gradient stochastique (LMS - *Least Mean Square*), l'algorithme du gradient stochastique normalisé (NLMS – *Normalize Least Mean Square*) et l'algorithme des moindres carrés récursifs (RLS – *Recursive Least Square*) ont été étudiés dans le cadre de ce mémoire. Du aux faiblesses (taux de convergence lent, complexité élevée dans le cas du RLS) inhérentes aux méthodes citées plus haut, une méthode d'identification robuste basée sur les algorithmes génétiques a été développée. Contrairement aux méthodes de filtrage classiques, la méthode basée sur les AG possèdent les avantages qui suivent : elle opère aussi bien dans les systèmes linéaires que non linéaires, elle est capable de procéder à l'identification de système même avec très peu de bit, possède une complexité moins élevée que le RLS en terme de "Full Adder".

Les résultats de simulations obtenus avec Matlab, pour les systèmes linéaires ont démontré que la méthode basée sur les AG était la plus performante, cela peu importe les grandeurs de bit utilisées. Des autres méthodes, seul le RLS a fourni des résultats à même de rivaliser avec les AG. Cependant afin de prétendre obtenir des résultats proches ou supérieurs à ceux des AG, le RLS doit se prévaloir d'une longueur de bit minimale de 24 bits. De manière générale nous avons pu constater que les AG ont besoin de 350 itérations normalisées ou 35 générations de population afin d'atteindre une erreur asymptotique stable. Nous avons également pu remarquer que Le LMS, le NLMS et le RLS éprouvent beaucoup de difficulté dans l'identification des paramètres b_1 et b_2. La répétitivité des AG, ainsi que celle des autres méthodes a été évaluée. Afin de vérifier que la méthode basée sur les AG n'est pas caduque, l'étude de la variance a permis de constater une excellente répétitivité des méthodes. Les AG ont démontré une grande robustesse, face à l'effet de quantification, en démontrant une variance quasi identique, peu importe les grandeurs bits employées. Au niveau du canal non linéaire, force est de constater l'inefficacité et l'incapacité des méthodes usuelles à identifier les paramètres du canal ARMA. Dans ce cas si, seuls les AG ont été opérationnels. Comme dans le cas linéaire, les AG sont d'une grande robustesse face à l'effet de quantification, ils sont capables d'identifier le système, même avec très peu de bits. Ils ont comme dans le cas linéaire besoin de 350 itérations normalisées ou 35 générations de population afin d'atteindre une erreur asymptotique stable. L'étude de la variance a permis de constater une excellente répétitivité de la méthode et une grande robustesse même avec très peu de bits.

Dans ce travail, une étude comparative des diverses méthodes a été effectuée. Cette étude nous a permis de comparer la complexité des AG avec celle du LMS et du RLS. En

terme d'opérations arithmétiques par itération et selon les paramètres des AG choisis (P = 10, $\beta = \alpha = 1$) nous avons pu constater que les AG sont plus complexes. Cependant, avec l'augmentation de la taille des filtres, la complexité du RLS croît beaucoup plus rapidement au point de surpasser celle des AG. En terme de "Full Adder" par génération, le RLS devient plus complexe, suivi des AG et du LMS. Nous avons également comparé les méthodes en fonction du nombre de "Full Adder" nécessaire pour atteindre un MSE de 3^{e}-2. Dans ce dernier exercice les AG étaient la méthode la moins complexe suivie du LMS et du RLS. De manière générale, nous avons noté qu'une augmentation de la dimension du filtre provoque irrémédiablement une augmentation de la complexité des méthodes.

Compte tenu des bons résultats obtenus par les AG sous MATLAB, la méthode a été implantée sous Simulink, dans l'optique d'une future conversion des blocs Simulink en bloc Xilinx afin de faire une synthèse de l'architecture. Cette synthèse nous permettrait d'estimer les ressources matérielles (nombre de additionneurs, de multiplieurs, de registres, de soustracteurs, de cellules logiques, de tampons, etc.) des AG sur les composantes programmables FPGA (*Field Programmable Gate Array*), afin de choisir le meilleur FPGA qui répond au compromis coût/performance. Les résultats obtenus dans Simulink vont dans le même sens que ceux obtenus dans Matlab et démontrent une fois de plus la robustesse de la méthode basée sur les AG qui fonctionnent pour les cas de systèmes linéaires comme non linéaires.

Durant notre étude, nous avons pu constater que les AG sont une alternative fiable en terme de qualité de résultats, de taux de convergence et de niveau de complexité vis-à-vis des méthodes de filtrage adaptatif usuel. Cependant, leur mise en œuvre nécessite beaucoup

de temps et d'ajustement des paramètres, afin d'obtenir ceux qui sont les mieux adaptés au problème.

Bibliographie

[1] F. Michaut, M. Bellanger, "Filtrage adaptatif", théorie et algorithmes, Paris :Lavoisier, 2005.

[2] V. Duong, A.R. Stubberud, "System Identification by Genetic Agorithm", IEEE aerospace conference proceeding, Vol. 5, 2002, pp. 2332-2337.

[3] M.S. White, S.J Flockton, "A Genetic Adaptative Algorithm For Data Equalization", IEEE world congress on computational intelligence, Vol. 2, 1994, pp 665-669.

[4] M. Kezunovic, Y. Liao " The use of Genetic Algorithms in Validating the System Model and Determining Worst-case Transients in capacitor Switching Simulation Studies", 9th Conference on Harmonics and Quality of power, Vol 2, 2000, pp. 685-690.

[5] Y-P .Sheng, L-S. Wang, "The Least Squares Solutions of Inverse Problem for Symmetric Orthogonal Anti-symmetric Matrices", IEEE Conference on Machine learning and Cybernetics, 2006, pp. 631-634.

[6] T. Söderström and P. Stoica, "On some system identification techniques for adaptive filtering," IEEE Trans. Circuits Syst, vol. 35, pp. 457–461,1988.

[7] G. P. Box and G. M. Jenkins, Time Series Analysis, Forecasting and Control, 3rd ed.Englewood Cliffs, NJ: Prentice-Hall,1994.

[8] M. Martone, "Blind deconvolution in spread spectrum communications over non minimum phase channels," in Proc. IEEE MILCOM Conf, 1994, pp. 463–467.

[9] Bazaraa, M. J. Jarvis, and H. Sherali, Linear Programming and network Flows, 2nd edition, Wiley, New York, 1990.

[10] S. Haykin. "Adaptive Filter Theory", Third Edition, Prentice Hall, 1996.

[11] C. Tannous, R. Davies, and A. Angus, "Strange attractor in multipath propagation," IEEE Trans. Commun., vol. 39, pp. 629–631, 1991.

[12] I. W. Hunter and M. J. Korenberg, "The identification of nonlinear biological systems: Wiener and Hammerstein cascade models," Biolog. Cybern, vol. 55, pp. 135–144, 1986.

[13] R. S. Ahmed. (1992, August), " Identification of Nonlinear Dynamic systems Using A Rapid Neural Network" 27th IEEE Conf of industrial electronics society , vol.3, 2001, pp. 1734-1739.

[14] M. Kozek, N. Jovanovic, " Identification of Hammerstein/Wiener Nonlinear Systems with Extended Kalman Filters" *American Control Conf,* vol.2, 2002, pp. 969-974.

[15] D. Massicotte, D. Eke, "High Robustness to Quantization Effect of an Adaptive Filter based on Genetic Algorithm", IEEE Int. Northeast Symp. Circuits and Systems **(NEWCAS)**, 5-8 Août, 2007.

[16] Y. Itoh, "High power Amplifiers for Mobile Communication Systems", in 1995 URSI International symposium, 25-27 0ct, 1995, pp 45-48.

[17] R.E. Kalman, "A new approach to linear filtering and prediction problems," Transactions of the ASME, Ser. D, Journal of Basic Engineering, 82, pp. 34-45, 1960.

[18] A. H. Jazwinski, Stochastic Processes and Filtering Theory. San Diego, CA: Academic, 1970.

[19] R. Amara, S. marcos, "A Blind Network of extended Kalman Filter For Nonstationary Channel Equalization", Acoustics, Speech, and Signal Processing Conference Proceeding, Vol.4, 2001, pp. 2117-2120.

[20] G. Glentis, P. Koukoulas, N. Kalouptsidis, "Efficient Algorithm for Volterra System Identification", Signal Processing Conference Proceeding, Vol. 45, 1997, pp. 1013-1024

[21] R. Boite, M. Hasler, H. Dedieu, " Effets Non linéaire Dans les filtres Numériques", 1ere édition, Presses polytechniques et universitaires romandes et CNET-ENST, Lausanne, 1997.

[22] J.H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, MI: The University of Michigan Press, 1975.

[23] D.H. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, New York: Addison-Wesley, 1989.

[24] K. De Jong, "A 10 Year Perspective," Proc. Int. Conf Genetic Algorithms and Their Applications, 1985, pp. 169.177.

[25] K. De Jong, "Learning with the Genetic Algorithm: An Overview" Machine Learning, vol. 3, pp. 121-137, Oct. 1988.

[26] W. Ying, L. Bin "Job- Shop Scheduling using Genetic Algorithm" IEEE international conference on systems and cybernetics, vol.3,1996, pp.1994-1999

[27] J.Zhang, Y. Zhang, R. Gao "Genetic Algorithm for optimal Optimal design for Vehicule Suspension" IEEE international conference on Engineering of Intelligent Systems, 2006, pp.1-6

[28] D.M. Etter, M.M. Masukawa, "A Comparison of Algorithms for Adaptive Estimation of the Time Delay Between Sampled Signals," Proc. ZEEE Znt. Conf ASSP, 1981,pp.1253-1256.

[29] D.J. Montana, and L. Davis, "Training Feedforward Neural Networks Using Genetic Algorithms," Proc. Int. Joint Conf Artificial Intelligence, Detroit, 1989, pp. 762-767.

[30] D.M. Etter, M.J. Hicks, and K.H. Cho, "Recursive Adaptive Filter Design using an Adaptive Genetic Algorithm" Proc. IEEE Int. Conf on ASSP, 1982, pp. 635-638.

[31] M. Gen, R.Cheng, "Genetic Algorithms And Engineering Optimization", 2000, pp.2-10.

[32] L. Ljung, "System Identification: Theory for the user", Prentice Hall, 1987.

[33] G. Liang, D. Wilkes, "The recursive Linear Method Identification for ARMA Method Estimation", IEEE transaction on automation control, 1992, pp. 677-682.

[34] W. Y, Wang, Y. H, Li, "Evolutionary Learning of BMF fuzzy-Neural Networks Using a Reduced-Form Genetic Algorithm", IEEE Transactions On Systems, Man, And Cybernetics , Vol. 33, No. 6, 2003, pp. 966-976.

[35] E-J. Ryu, J-W. Lee, J. Lee, E-S. Cho, "Signal Quality Measurement Using Channel Identification Method for High-Density Optical Channel", IEEE transaction on Magnetics, 2005, pp. 977-979.

[36] T. Kinnunen, E. Karpov, P. Franti Liang, D. Wilkes, "Real-Time Speaker Identification and Verification", IEEE transaction on Audio, Speech and Language Processing, 2006, pp. 277-288.

Annexe – Publication

D. Massicotte, D. Eke, "High Robustness to Quantization Effect of an Adaptive Filter based on Genetic Algorithm", IEEE Int. Northeast Symp. Circuits and Systems (**NEWCAS**), 5-8 Août, 2007.